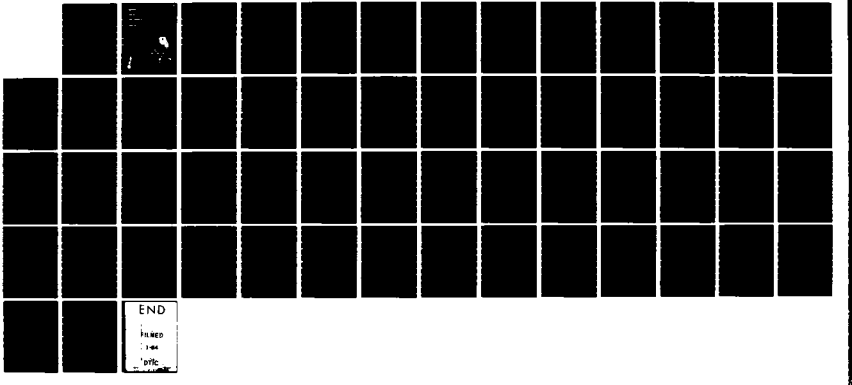
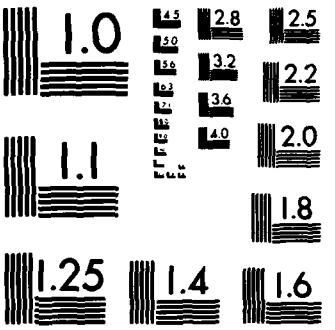
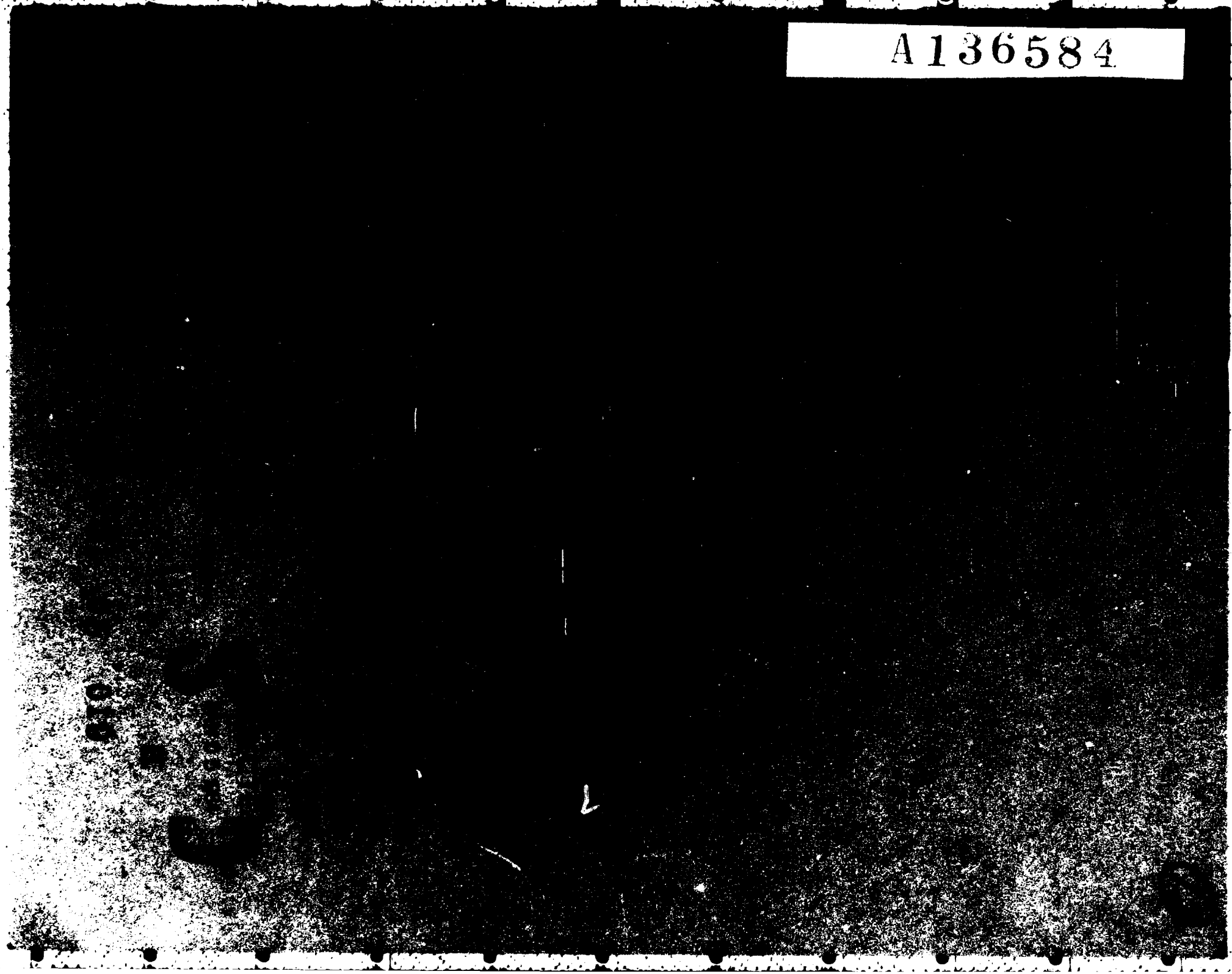
AD-A136 584 WRITING TSO (TIME SHARING OPTION) COMMANDS IN FORTRAN (U) HARRY DIAMOND LABS ADELPHI MD A HAUSNER OCT 83 HDL-TM-83-20 1/1

UNCLASSIFIED F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER HDL-TM-83-20	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Writing TSO Commands in Fortran		5. TYPE OF REPORT & PERIOD COVERED Technical Memorandum
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR*(s)* Arthur Hausner		8. CONTRACT OR GRANT NUMBER*(s)*
9. PERFORMING ORGANIZATION NAME AND ADDRESS Harry Diamond Laboratories 2800 Powder Mill Road Adelphi, MD 20783		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Materiel Development and Readiness Command Alexandria, VA 22333		12. REPORT DATE October 1983
		13. NUMBER OF PAGES 64
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

HDL Project: Y91EY1

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

TSO
Commands
Fortran
IBM
Productivity

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

By having various assembler routines interface facilities not normally available in Fortran, TSO (time sharing option) commands are easily written in Fortran. Major interface routines written at Harry Diamond Laboratories are described in detail, with some helpful hints for using them. Minor routines, including some written in Fortran, are briefly described. Some general useful techniques for writing commands are also presented. Finally, three examples are given to illustrate the techniques: (a) SYSOUT, written to provide a copies keyword for a sysout file, (b) READN, to inhibit printing a user's response to a read (for, say, a password), (c) COMPRESS, for compressing a PDS (partitioned data set) in place.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

1 SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

CONTENTS

TABLES

1. INTRODUCTION

The Harry Diamond Laboratories (HDL) Computing Center, consisting of a mainframe IBM 370/168 computer operating under multiple virtual storage (MVS), has had considerable success writing TSO (time sharing option) commands in Fortran. All that is required are a few basic assembler routines to interface various facilities normally available only to assembly programmers. With this capability, the usual advantages of writing programs in a high-level language accrue: rapid evolution of programs, easy debugging, and easy maintenance. These features more than offset the major disadvantages: more execution time and more memory are required than for assembler programs.

The facilities to be described were slowly evolved over a period of several years. They were used at first to help speed up the many system command lists (Clists) that had been written, by having entire sections of Clist code replaced by a command to perform the equivalent functions. Before too long, however, it was recognized that the entire Clist could be replaced by a Fortran command provided certain interfaces were available; the Management Information Systems (MIS) Office* began to develop these routines for that purpose. It was noted that large amounts of execution time and resources could be saved as a result. A typical example, that of a foreground compile of a Fortran program, reduced the time to call the compiler by a factor of 4; and since allocations were made with the convertible attribute (to be described) and not unallocated, a second use of this command in the same TSO session resulted in even greater savings.

The techniques to be described here are not limited to systems programmers installing system commands for the user community. Any load module can be executed as a command in a Clist using the TEST command:

```
TEST  loadname  CP
(command syntax)
RUN
```

or the HDL-written command, CMD:

```
CMD loadname =(command syntax)
```

The resulting execution will generally be much faster than an equivalent Clist.

Most commands have the following general structure:

SUBROUTINE statement
COMMON, TYPE, and DATA statements
PARSE statements
ATTEN statement for attention handling
Determination of which keywords are used, and performing tasks based on their values. These include dynamic allocation of files, interactions with the user, calling other commands, and attaching other programs.
Cleanup statements
QUIT statement to provide a return code
END

*The assembler routines were written by B. H. Audet of the MIS office, and Jane Schmitt, William Jacob, and Walter C. Bodycomb of IBM.

2. THE MAJOR INTERFACE ROUTINES

The most important assembler interface subroutines that have been written are the following:

ATTACH—Executes any program from any other program, including a command.
ATTEN—Provides attention handling.
COMAND—Executes any command from another command and optionally receives alpha information from QUIT in the called command.
DYNALC—Interfaces SVC99, the dynamic allocation services.
PARSE—Interfaces the TSO parser.
QUIT—Provides a return code to the caller and optionally transmits alpha information to a calling command.
TSPTGT—Interfaces the PUTLINE, GETLINE, and PUTGET facilities for communicating with the terminal.

Appendix A describes these routines and their calling sequences, but some notes on using the routines are described in sections 2.1 through 2.7. The notes should be read with appendix A as a reference.

2.1 *Subroutine ATTACH*

ATTACH permits any program to be executed from any other program or command. It is useful for calling compilers, utilities, or user-written programs.

When using ATTACH, it is generally useful to have the first argument, IOPT, set to 1, to enable an attention to be intercepted by the command, for cleanup operations and to set a proper return code.

The program being attached does not have to be in an automatic call library. Before calling ATTACH, allocate the library without a member by calling DYNALC using verb code 1 and key 85 to allow the system to select a ddname, which is returned to the program. It will be an 8-character name beginning with SYS0. The argument PGM in ATTACH is then specified with the 16 contiguous bytes consisting of the ddname followed by the program name. To be sure the program exists as a member in the library allocated, subroutine MEMCHK (see sect. 3) may be called as an intermediate step.

The PARMs that may be entered must be interpreted by the program being attached. Compilers, for example, are generally written to allow two sets of parameters: the first to define options to the compiler, and the second to define a set of nonstandard ddnames to be used in place of standard ones.

2.2 *Subroutine ATTEN*

ATTEN provides attention handling for any program or command. The routine is called at the beginning of the program or command. If attention is signalled by the user an attention argument is set. It must be periodically checked to see if attention was signalled. The user may cancel attention control at any time by calling ATTOFF and reinstitute attention control by calling ATTEN again, a useful feature for commands that have subcommands.

Clist ATTN statements have priority over this routine. If the program or command is called from a Clist, ATTN OFF should be in effect for the ATTEN subroutine to be recognized. If two (or more) interrupts occur before the system responds to the first one, the program or command will give up control to the TSO processor.

Note the important restriction in the description of ATTEN (app A). An optimizing compiler that rearranges logic has no way of knowing the attention argument may be changed from outside the program. To prevent shifting of statements, the argument should be placed in a COMMON statement. For example, the statements

```
      LOGICAL AFLAG
      ..............
      AFLAG=.FALSE.          (initialization)
      CALL ATTEN(AFLAG)
      ..............
      DO 10 I=1,N
      IF (AFLAG) GOTO 20
      CALL DYNALC( etc.  )
      ..............
10    CONTINUE
      ..............
20    CALL QUIT(LC)
```

will result in the optimizer removing the statement checking AFLAG from the loop, since AFLAG is not modified in the loop. (A really smart compiler will remove it altogether because it is not changed after its .FALSE. initialization.) To prevent this, add the statement:

```
COMMON /ATTN/AFLAG
```

Now the compiler cannot know the DYNALC does not have the COMMON block /ATTN/, and hence it assumes that AFLAG may change in the call to it. Therefore, the check on AFLAG will not be relocated.

ATTEN should be called after the parsing statements (see subroutine PARSE) in a command, to permit a user to exit a command in response to a parsing prompt.

2.3 *Subroutine COMAND*

COMAND executes a command from another command, and transmits character information back to the calling command if the called command supplies information (in an attempt to simulate the Clist GLOBAL statement).

One blank is automatically supplied between the command name and the first keyword, but additional spaces between keywords must be supplied as part of the CMDRGS. The example in the description (app A) illustrates this procedure.

Note that some commands do not work (the CALL and EXEC commands, for example).

2.4 *Subroutine DYNALC*

DYNALC is an interface to SVC99, to make dynamic allocations, unallocations, concatenations, deconcatenations, and retrievals of information concerning existing allocations.

Much of the information contained below is meaningful only when supplemented with the information in the IBM manual GC28-0627, OS/VS2 System Programming Library: Job Management.[1] Because of the complex nature of dynamic allocation, several Fortran subroutines were written to perform certain allocation functions. All are described in detail in 'SYS1.PAGCAT' and easily accessible with the PAGCAT command. The routines are listed in table 1. More routines will be added as the need arises.

TABLE 1. FORTRAN SUBROUTINES DESIGNED FOR SPECIFIC ALLOCATION FUNCTIONS

Fortran routines	Purpose
ALCMSG	Provides an allocation error message. (Routine DYNALE is more general.)
ALLC	Allocates an existing dataset, performing checks generally made in commands and supplying error messages.
ALLCD	Allocates a dummy file.
ALLCT	Allocates a terminal file.
ALLCTD	Allocates a terminal file with a DCB.
ALLCW	Allocates a temporary work file.
ALLCWD	Allocates a temporary work file with a DCB.
CONCAT	Concatenates existing datasets.
CONCTV	Concatenates existing datasets when volumes are known
CREATE	Creates a new dataset.
DELETE	Deletes a dataset with dynamic allocation. (Subroutine DSSCR does not use dynamic allocation and is more efficient.)
FREE	Unallocates a file, or marks it not in use.

It is best to choose names for the texts (see app A) to correspond to the requested functions, to make it simpler to interpret. For example:

```
CALL DYNALC (1,IER,IERR,INFO,0,DSIN,DDPRT,PWD,MEM,SHR,PERM)
```

contains the usual error return codes IER, IERR, and INFO, and supplies dynamic allocation texts DSIN, DDPRT, PWD, MEM, SHR, and PERM. Each text is defined in a DATA statement that specifies the key, #, LEN, and PARM. (LEN and PARM are omitted if # is 0; LEN and PARM are repeated # times if # > 1.) Because key, #, and LEN are all 2-byte integers, it is most convenient to use INTEGER*2 as the type. Thus, the above example might have DSIN look like

```
INTEGER*2 DSIN(8)/2,1,10,'SY','S1','.P','PL','IB'/
```

where

key = 2 (defines a dataset name),
# = 1 (one name to be defined),
LEN = 10 (10 characters in the name), and
PARM = SYS1.PPLIB (the name).

Various keys to define allocation attributes are defined in the aforementioned IBM manual, GC28-0627.[1]

The above example uses DS as part of the dataset name text to suggest that it is just that. All the names used suggest that a dataset name specified by DSIN is to be allocated with a member (MEM text) using a ddname specified by the DDPRT text, with a password specified (PWD text), status SHR, and using the permanently allocated attribute (PERM text). In this way, the statement is more readable and easier to debug and maintain.

[1] *"OS/VS2MVS System Programming Library: Job Management," IBM Manual GC28-0627, IBM Corporation, Poughkeepsie, N.Y.*

NTEXTS=0 (where NTEXTS is normally the number of texts) can always be used in a general allocation call. This forces DYNALC to check all texts supplied. Texts may be ignored by making the key=0 if they are not wanted. For example, in the above call, MEM and PWD may be defined by:

```
INTEGER*2 MEM(7)/3,1,5*' '/,PWD(7) 80,1,5*' '/
```

with the member length, MEM(3), and value MEM(4)-MEM(7), as well as the password length, PWD(3), and value PWD(4)-PWD(7) stuffed into the positions if the user specifies them in the command syntax. If he does not specify either or both, the program can assign MEM(1)=0, or PWD(1)=0, or both. Thus, the call to DYNALC is still valid.

Although INTEGER*2 is the most convenient specification for texts, it is not necessary. For example, to specify units of space in tracks, we may use

```
INTEGER*2 TRACKS(2)/7,0/,       (or)
INTEGER*4 TRACKS/Z00070000/,    (or)
INTEGER*4 TRACKS/1792/
```

In all cases, the four consecutive bytes on TRACKS contain hex 00070000, defining key=7 and #=0.

Several allocation features allowed under JCL are not allowed dynamically:

(a) DDnames of JOBCAT, STEPCAT, JOBLIB, and STEPLIB,

(b) Keyword parameters AMP, BURST, CHARS, CHKPT, DDNAME, DLM, DSID, FLASH, and MODIFY,

(c) Positional parameters *, DATA, and DYNAM,

(d) Selected subparameters of keywords:

Group values of COPIES,
CYLOFL, NTM, and RKP of DCB characteristics, as well as a ddname reference to a previous step,
PASS of DISP,
Reference to ddname ISAM area name of DSN,
ABSTR specification of SPACE,
DEFER specification and AFF of UNIT,
RETAIN specification and REF=ddname of VOLUME,

On the other hand, several allocation features not allowed under JCL are allowed dynamically (for IVERB=1), as shown in table 2.

Of the above, the permanently allocated and convertible attributes require clarification, because the concepts do not exist for JCL allocations.

The permanently allocated attribute is used by the TSO ALLOCATE command and all allocations made by JCL for batch jobs. Besides other features, this means that the system can-

not unallocate the resource to satisfy another allocation request to meet the control limit (set by DYNAMNBR of the logon procedures—75 using most HDL logon procedures).

TABLE 2. ALLOCATION FEATURES ALLOWED DYNAMICALLY BUT NOT ALLOWED IN JCL

Key (decimal)	Feature
80	Password specification
82	Permanently allocated attribute
83	Convertible attribute
85	Ddname return specification (if you let the system select a ddname)
86	Dsname return specification (if you let the system select a dsname)
87	DSORG return specification
93	Volume serial return specification

The convertible attribute means that the system has some flexibility to reuse the allocation if it is marked not in use (either by verb code 2, key 8, or because a command ends). The system will not reuse the allocation again unless only certain parameters are different from the new request. These are: ddnames, member name, status, normal and conditional dispositions, space unallocation at close, input and output only, DCB attributes, password, and the permanently allocated attribute (which may also be specified at the same time). In addition, any allocation not in use that does not have the permanently allocated attribute is eligible for automatic unallocation to satisfy the control limit.

A good practice is to allocate with the convertible attribute (key 83), letting the system select a ddname (key 85) whenever possible. Utilities and compilers have provision for specifying ddnames other than standard ones. Existing allocations with the convertible attribute then will be used by the system, if possible, thereby saving much processing. Such allocations need not be freed at the termination of the command, so they may be reused if needed. However, datasets that may be required for exclusive use in a batch mode, or by other users, should be freed.

An existing allocation that does not have the convertible attribute cannot be chosen for an allocation specifying it. Therefore, allocate specific ddnames (SYSIN, SYSPRINT, etc.) with the permanently allocated attribute and without the convertible attribute. In this way, a system-selected ddname will always begin with "SYS0" and contain eight characters; no conflict will arise in any selection.

Use both the convertible and permanently allocated attributes for dummy and terminal files with system-selected ddnames. Without the permanently allocated attribute, they are automatically unallocated when marked not in use (at the conclusion of the command). The same is true for work files for which no dsname is specified.

An existing allocation may be reused even if it has only the permanently allocated attribute. Thus, if you specify the equivalent of

```
ALLOC DD(SYSPRINT) DS(*)
```

and SYSPRINT is already allocated to the terminal, the allocation will be successful. For this reason, a "REUSE" keyword should be simulated as follows, with a specified ddname:

(a) Attempt allocation,
(b) If successful, go to (h),

(c) If unsuccessful, and the reason code is 1040 (ddname in use), go to (e),

(d) Error message and terminate,

(e) Free ddname,

(f) If successful go to (a),

(g) Go to (d),

(h) Allocation successful—continue.

In this way, an existing allocation is used even if it only has the permanently allocated attribute, but only if it is not in use and no parameters need be changed. The alternate way (free ddname and allocate ddname to the terminal), may free an existing allocation and is less efficient.

Deconcatenate concatenated datasets (using IVERB=4) in a cleanup operation before command termination, to make the component allocations available for future use.

Specify the volume (key 16) in a dsname allocation. If not supplied by the user (or command), obtain the volume first from the catalog with subroutines DSORG, RECFM, LOCATE, or LOCAT6 (see sect. 3). This protects users who work with cataloged and uncataloged datasets with the same name. If the user allocates the uncataloged dataset first by specifying a volume, and if the cataloged dataset without volume is allocated second, the system gets the volume for the second allocation from the first allocation and *not* from the catalog. This is not what the user wants to happen.

2.5 *Subroutine PARSE*

PARSE allows command writers to use the major elements of the TSO parser, including prompts, error detection, recognition of mutually exclusive keywords, dataset names with members and passwords, keyword parameter abbreviations, etc. It is the most difficult of all the interfaces to describe, because of the additional seven entry points:

PARDSN—specifies name(s) of dataset(s), with member(s) and password(s) for a parameter.

PARPOS—specifies character value(s) of a parameter.

PARDEC—specifies binary (integer) value(s) of a parameter.

PARSTR—specifies a string value for a parameter.

PARKEY—identifies the variable to change if a keyword is used, i.e., defines a keyword.

PARNAM—specifies keyword names.

PARSUB—identifies the keyword for which the next call to PARDSN, PARPOS, PARDEC or PARSTR applies.

The order of the calls to the different entry points is important. In the following we use the notation VALUECALL for any of PARDSN, PARPOS, PARDEC, or PARSTR. The order is as follows:

(a) One VALUECALL for each positional parameter of the command.

(b) One PARKEY call for each set of mutually exclusive keywords, followed by a set of PARNAM calls defining the keyword names, and if the keyword has a value, an identifying number.

PARKEY and a set of PARNAM are repeated until all keywords have been specified. The examples in appendices A through D should clarify this procedure.

(c) A PARSUB call with the identifying number specified in PARNAM above, followed by one or more VALUECALLS to set the values. PARSUB and VALUECALLS are repeated until all values are accounted for.

(d) PARSE, which sets all the values. The crucial values are
IWHICH (of PARKEY) to tell the command which keywords were specified by the user.

The first argument of PARDSN, PARPOS, PARDEC, and PARSTR to tell the command values of keywords specified, if those keywords have values.

All items are not necessarily involved. If there are no keyword parameters, items (b) and (c) are not needed. If no keywords have values, item (c) is not needed. If there are no positional parameters, item (a) is not needed. The use of these routines will become clearer during the discussion of examples in section 5.

The string entries in the calls to various entry points are self-defining strings in which the first and last characters are identical and the character is not used within the string. PARSE strips away those two characters, leaving the remaining as the string entered. Thus, no length variables need be supplied for the strings.

Length variables are needed for values that users supply, and are available in the argument LENS (see app A) for all entry points. By using IOPT > 15, the lengths are assumed INTEGER*4 variables. This is generally more convenient because the use of other subroutines (e.g., for error messages) requires INTEGER*4 variables. However, INTEGER*2 is sometimes useful for stuffing the value directly into the length field of a dynamic allocation text.

If "*" is entered by the user in a request for a dataset name (subroutine PARDSN), the dataset prefix is not left-appended to it, even if requested, and its length is returned as 1.

When using entry PARPOS for obtaining values of keywords, the combination of MAXL, IFIRST, and IOTHER defines the maximum number of characters to be entered, the type of the first character (any, alphabetic or national, numeric, alphabetic only, and alphabetic or numeric only), and the type of the other characters. Combinations are shown in table 3 below with the numeric designations as described in appendix A. The JCL manual usually defines legal values for other parameters (FORM name, FCB, etc.)

TABLE 3. COMMON COMBINATIONS OF CHARACTERS

Entry	MAXL	IFIRST	IOTHER
Ddname	8	1	3
SYSOUT destination	8	3	3
Disk volume*	6	3	3
Clist variable name	31	4	5
Member name	8	1	3
Password	8	3	3

*Volume characters can include a hyphen and other special characters, but these designations are not used at HDL.

2.6 *Subroutine QUIT*

QUIT provides a command with a return code that can be checked with &LASTCC in a Clist. If, however, the command is called by another command, QUIT provides the return code to the calling command, and can also provide character information to that command. This is useful to prevent recalculation of important variables. For example, the COBOL compile command, CVC, provides the name of the object module to the compile and link command, CVCL, which then can more easily call a link command. The entire process is an attempt to simulate a Clist GLOBAL statement.

The address of the block of character information is stored in the register normally reserved for return codes, which are useful in Clists to control the flow of commands by checking LASTCC. For normal operation from a Clist or from TSO, the return code should be in this register. A procedure was devised to insure that the return code is in the register if the command was not called from another command as follows.

Commands that are expected to return character information to a calling command are given an alias (CVC# in the above example). Since the name by which a command is called can be picked up by the parser (subroutine PARSE has an optional argument for this), it is a simple matter to determine which form of the QUIT subroutine to use. The alias name is always used when a command is called by another command, and it is expected to return information to that command. Thus, the call

```
CALL COMAND(3,IRC,'CVC#     ',RTNLEN,RETARG, ... )
```

alerts the CVC code that return character information is expected. The CVC code has

```
LOGICAL*1 RETARG(174)
INTEGER*4 LARG/174/
REAL*8 CMDNAM,NOPASS/CVC#/
.....................
CALL PARSE(IER,CMDNAM)
.....................
IF (CMDNAM.NE.NOPASS) LARG=0
CALL QUIT(IRC,LARG,RETARG)
.....................
```

Hence, LARG is not set to 0 for the call to QUIT, so the information is returned to the calling command. With this code, if CVC# were called in a Clist,

```
CVC# dsname
```

the return code for checking LASTCC would be incorrect.

2.7 *Subroutine TSPTGT*

TSPTGT allows commands to make nearly full use of the PUTLINE, GETLINE, and PUTGET facilities to write and receive messages from the terminal. This includes

(a) Message identification (MSGID) that is subject to the MSGID/NOMSGID characteristics of the user's profile. HDL commands have not used this feature.

(b) Multilevel messages, each of which will be followed by a plus ("+"). A higher level message is displayed if the user responds with a question mark ("?").

(c) Capability to fold input or output to upper case or leave it as is.

(d) Write or receive messages subject to a MSG/NOMSG and PROMPT/NOPROMPT Clist CONTROL environment.

(e) Inhibit printing the response of a user PUTGET message.

(f) Inhibit a carriage return after printing a message.

(g) Sound the audible alarm (BEEP), subject to MSG/NOMSG Clist CONTROL, or not.

Item (e) is available only in the PUTGET facility, which requires a CONTROL PROMPT to be active in the Clist. For all other situations, it is generally more convenient to use the PUTLINE and GETLINE to write a message and receive an answer. Two subroutines were written to simplify the calls to TSPTGT:

TSPUT—an assembler routine to interface the PUTLINE facility (by calling TSPTGT).

TSGET—a Fortran routine to interface the GETLINE facility.

These routines have two fewer arguments than TSPTGT and are easier to use.

The facility uses the self-defining string notation described by PARSE for receiving strings without the length specification. However, a length and a string (two arguments) may be optionally used and both may be intermixed in one call. Messages can be built up on several calls by terminating the call with the argument −1. For example, the calls

```
      CALL TSPTGT (IRC,1,0,'/''/', LDS, DS, -1)
      IF (IMEM.NE.0)
     *          CALL TSPTGT (IRC,1,0,'/(/', LMEM,MEM, '/)/',-1)
      CALL TSPTGT (IRC,1,0,'/'' WAS GENERATED/')
```

insert a member name, MEM (with length LMEM), with parentheses in the message (LDS is the length of DS, the dataset name):

```
'dsname' WAS GENERATED
```

to become

```
'dsname(member name)' WAS GENERATED
```

if the member name exists (IMEM.NE.0). Thus, the facility is very user-oriented, not requiring stuffing of lines, keeping track of line lengths, etc. The third argument determines whether or not beeps are sounded, whether MSGID's are included, whether the message is subject to a Clist CONTROL NOMSG/MSG environment, etc.

The GETLINE facility does not require CONTROL PROMPT to be active in a Clist where the command is used. In general, the prompting is always desirable, so that GETLINE is considerably more advantageous than PUTGET, which does require the CONTROL PROMPT. The most common usage would be

```
                    LOGICAL*1 AREA(20)

    CALL TSPTGT(IRC,1,80,'/ENTER DATA/')
or
    CALL TSPTGT(IRC,1,112,'/ENTER DATA: /')

(to inhibit carriage return), followed by

    LEN=20
    CALL TSPTGT(IRC,2,40, AREA,LEN)
    IF (LEN) 10, 20, 30
C RETURN IS NEGATIVE
10  IF (LEN. EQ. -9) GOTO ...    (attention returned)
    IF (LEN, EQ. -8) GOTO ...    (more than 20 characters
                                  were entered)
    (write error message)
    GOTO ...                 (transfer to execution after error)
20  ...                 (CARRIAGE returned only, no data input)
    ...
    GOTO ...
30  ...   (1-20 characters were entered and are stored in AREA)
    ...
```

In the above, the 80 in the TSPTGT call sounds an alarm and treats the string as data (it is printed in a Clist even if CONTROL NOMSG is active). The 40 in TSPTGT folds the input to upper case and gets the input from the terminal. In a Clist, the call to the command using this code could be followed by

```
DATA PROMPT
(response)
ENDDATA
```

even if CONTROL NOPROMPT is in effect. If not followed by the DATA PROMPT statement, responses will be expected from the terminal.

3. OTHER USEFUL ROUTINES

Other subroutines were developed as they were needed, even for esoteric applications. Table 4 describes the name of each routine, the language each is written in, and its purpose. All of these routines have catalog files in 'SYS1.PAGCAT', which may conveniently be listed with the commands PAGCAT or PAGCATFS (a full-screen version) on TSO.

TABLE 4. OTHER USEFUL SUBROUTINES

Name	Language	Function
ABEXIT	Assembler	Sets up an error exit for system abends. There is also an entry point, ABCNCL, to cancel it.
ACFRET	Assembler	Returns information from the user's ACF logonid record.
ACTIVE	Assembler	Determines whether a given job name is currently executing. Used mostly to see if the System 2000 multiuser job is active.
ACTNO	Assembler	Returns the user's account number.
BEEP	Assembler	Sounds an audible alarm of 3270-type terminals.
BEEPC	Assembler	Sounds an audible alarm of 3270-type terminals, subject to the Clist MSG/NOMSG environment.
BITFL	Assembler	Flips a bit.
BITN	Assembler	Sets a bit ON.
BITOF	Assembler	Sets a bit OFF.
BITSET	Assembler	Sets a bit ON or OFF.
CATINF	Assembler	Returns catalog information. Later superseded by the faster CTINF.
CHFIL	Assembler	Fills an area with a character.
CHMOV	Assembler	Moves a string from one area to another.
CKDSQ	Assembler	Returns users/jobs currently allocated or enqueued to a given dataset and their statuses.
COMBLD	Fortran	With entry points COMBLN and COMQUO. Rebuilds a syntax line for use in calling one command from another in subroutine COMAND.
CTINF	Assembler	Returns catalog information: dataset names, volumes, and units.
C3270	Assembler	Clears a 3270-type screen.
DATE	Assembler	Returns today's date in the form MM/DD/YY.
DATEB	Assembler	Returns a date in the form 07 JUL 1981.
DATEJC	Fortran	Converts a Julian date string to the form MM/DD/YY.
DESTCK	Assembler	Checks the validity of a remote destination. Useful to check user's entry of a destination.
DFDCB	Assembler	Returns the editor default DCB specifications for a dataset type. Useful to determine these from a name only.
DISKSL	Fortran	Finds the user disk (USER01-15) that contains the largest number of free contiguous cylinders. Aids in disk selection.
DMPFMT	Assembler	Formats a line of dump data.
DRCTRY	Fortran	Determines the number of directory blocks allocated to a PDS, and the number used. However, it is more efficient to use MEMRED if done while reading member names of a PDS.
DSATTR	Assembler	Returns selected VTOC information for a dataset.
DSCB	Assembler	Returns all VTOC information for a dataset
DSKDEV	Fortran	Determines the device type for a given mounted disk.
DSKLST	Fortran	Selects from an input list the name of the disk with the largest number of contiguous free cylinders available. Uses SRO010.
DSORG	Fortran	Returns a number from which the DSORG of a disk dataset can be found. Uses DSATTR and contains that output in common. Also provides error messages.
DSSCRU	Assembler	Scratches, uncatalogs, or both scratches and uncatalogs a dataset.
DYNALE	Assembler	A routine callable after a failed allocation with DYNALC, interfacing the DAIRFAIL program for error messages. Uses the PUTLINE facility.
DYNALM	Assembler	A routine callable after a failed allocation with DYNALC, interfacing the DAIRFAIL program for error messages. The error messages are not printed, but stored in the calling program for disposition by the user.
DYNALW	Assembler	A routine callable after a failed allocation with DYNALC, interfacing the DAIRFAIL program for error messages. Uses the WTP (write-to-program) facility. Batch users will find such messages in the JES-2 job log.
EDITX	Assembler	Converts an integer*4 value to character format, right-justified, and blanks leading zeros.
EDITZ	Assembler	Converts an integer*4 value to character format, right-justified, and inserts leading zeros.
ELTIME	Assembler	Computes elapsed CPU or task time.
FASTIO	Assembler	Performs unformatted I/O to any sequential file with fixed, variable, or undefined length records. This routine is the bulkwark read and write routine to or from files. It is considerably more efficient than Fortran reads and writes and its compatibility across record formats makes it more convenient in many applications.
FCVxxx	Assembler	A series of conversion routines that links with routines in the Fortran library.
FOLDUP	Assembler	Translates character data to upper case.
FSORT	Assembler	A sort interface with the SORT/MERGE utility.
FSxxxx	Assembler	A set of full screen routines for use in creating menu-driven programs and commands. Useable only on 3270-type terminals and, as such, has limited use for general-purpose commands.
GETCSV	Assembler	Gets the value of a variable from a Clist.
GETPRM	Assembler	Supplies either the date, time, userid, dataset prefix, logon procedure name, or default sysout destination.
GETUCB	Assembler	Describes the address, name, device type, mount attribute, use attribute, and allocation status for each disk in the system, and the total number of disks.
GTSIZE	Assembler	Returns the logical terminal line length and the number of rows if the terminal is a screen.
IBIT	Assembler	Tests a bit in a variable to see if it is set or not.
ICSORT	Assembler	Performs a shell sort of an array of records stored in memory.
IDAY	Assembler	Determines the day of the week for a given day.

TABLE 4. OTHER USEFUL SUBROUTINES (Cont'd)

Name	Language	Function
JOBCHR	Assembler	Obtains an alphabetic character (and its corresponding number from 1 to 26) for use in names
JOBINF	Assembler	Returns information about a user-supplied job name: the number (or numbers) and its status. Commands using JOBINF must be linked with AC=1 and the command name must be in the authorized command list, CSECT IKJEFTE2 of load module IKJEFT02.
JOBNO	Fortran	Calls JOBINF and supplies the job number of the most recently submitted job. This routine uses a specific algorithm if there is more than one job number for a given job name. It is useful for supplying the user the job number for a batch job written directly to the internal reader.
JOBTYP	Assembler	Determines if current command is being processed in foreground, background, or as a console started task. For commands, however, it is usually just important to determine foreground or background. Since the default destination of a background job is null, this information is deducible from routines TSUDF1 or GETPRM.
JULIAN	Assembler	Provides today's Julian date.
LOCATE	Assembler	Locates the volume on which a given dataset resides.
LOCAT6	Assembler	As in LOCATE, but the volume argument requires 6 bytes instead of 8.
LSCAN	Assembler	Scans a specified character string for a given substring.
LSTCM	Assembler	Logical string comparison.
MEMCHK	Assembler	Checks a given PDS to see if it contains a given member.
MEMDEL	Assembler	Deletes a member of a PDS.
MEMINF	Assembler	Returns information about members in a PDS: names, lengths, and TTR's.
MEMRED	Fortran	Also returns the information in MEMINF, but by reading the directory one name at a time. It is not necessary, therefore, to dimension the names, lengths, and TTR's: large datasets can be processed. MEMALC (to allocate the PDS) and MEMCLS (to end reading) are entry points.
OPEN	Assembler	Opens and closes a dataset. Useful to provide an end-of-file mark or to process an RLSE feature after allocating MOD.
PASSCK	Fortran	Determines the password protection status of a dataset. Similar to DSORG.
RDVTOC	Assembler	Reads the VTOC of a disk, one dataset at a time.
RECFM	Fortran	Returns a number from which the record format of a dataset can be found. Similar to DSORG.
RECMSG	Fortran	Displays an error message after RECFM returns an unwanted number.
REGSZ	Assembler	Returns the logon region size in bytes.
RENAM	Assembler	Renames a dataset.
SETCSV	Assembler	Sets a defined Clist variable to a new value.
SRO010	Assembler	Finds the amount of free space remaining on a specified DASD volume.
STGxxx	Fortran	A set of routines to perform string manipulation including conversion of character to numeric data and vice versa, blanking, moving, etc. These routines use CHFIL, CHMOV, LSTCM, and LSCAN when possible, for efficiency, and to provide error checking.
SUBMT	Assembler	Writes a file to an internal reader.
TIME	Assembler	Provides the system time of day in the form HH:MM:SS.
TIMEB	Assembler	Provides the system time of day in the form HH:MM:SS.TH.
TPIO	Assembler	Reads from and writes to a terminal. Unlike TSPTGT, this routine cannot be subjected to a Clist CONTROL MSG/NOMSG environment and was used prior to that routine. (TPIO can be used in a program not executed as a command).
TPUTAS	Assembler	Writes to the terminal without a carraige return, and can be used in programs not executed as a command. For a command, TSPTGT is more versatile.
TSGOPN	Assembler	and its entry points TSGGET, and TSGCLS—A set of routines that enable reading from allocated files that need not be Fortran files. Very similar to FASTIO.
TSHEX	Assembler	Converts data to hexidecimal format suitable for use in routines that use self-defining delimiters (such as TSPTGT or TSMSG).
TSINT	Assembler	Converts INTEGER*4 integers to character format with self-defining delimeters, suitable for use in TSPTGT or TSMSG.
TSMSG	Assembler	Writes a message to the terminal. Unlike TSPTGT, it cannot be subjected to a CONTROL MSG/NOMSG environment.
TSPFLD	Assembler	Retrieves the parameters of the command. This routine is useable if you wish to do your own parsing in the command (instead of subroutines PARSE). It takes much less memory but just presents the parameters as specified It is useful if there is one keyword.
TSPOPN	Assembler	With entry points, TSPLEN, TSPPUT, and TSPCLS writes to an allocated file. Very similar to FASTIO, with the added advantage that it uses self-defining delimiters to make it easy to build up messages over several calls. If the LRECL is not defined, TSPOPN also sets DCB values (LRECL=80, BLKSIZE=3120, and RECFM=FB).
TSUDF1	Assembler	Finds the userid, prefix, and default destination of the user. If the userid is null (as it would be in a batch mode), it is set to the account number. If the prefix is null, it is set to the userid. Similar to the functions of GETPRM
UNITS	Assembler	Returns a list of DASD names. Can return all names, only on-line names, public device names, or storage device names.
WAIT	Assembler	Places the active task in a wait state for the specified number of microseconds. Useful to pause a while before retrying an allocation.
WTO	Assembler	Writes a line of text to the operator's console.
WTOR	Assembler	Writes a line of text to the operator's console and waits for a reply
ZEDIT	Assembler	Converts binary data to hex.

4. USEFUL TECHNIQUES

Many useful techniques evolved while the many commands that now exist in the HDL computer environment were being written. Some have already been described during the discussion of major interfaces in section 2. Below are listed additional techniques found useful.

(a) Add a SUBROUTINE statement with no arguments at the beginning of the command. This eliminates the need for having file FT06F001 allocated for execution errors. Without the SUBROUTINE statement, users who do not have file FT06F001 allocated cannot execute the command.

(b) Decide carefully on the syntax of the command, using keyword names to suggest their functions. It is also useful to try to arrange names to take advantage of the abbreviations the parser permits. For example, two keywords called KEYWORD1 and KEYWORD2 would have to be completely spelled out by the user to avoid ambiguity. There are usually alternate names that can be used to allow fewer user keystrokes. Keywords with a few possible values should be split into mutually exclusive keywords without value (START/NOSTART instead of START (YES or NO)). Occasionally, both forms might prove convenient. In the HDL PR command the sysout class was represented by

```
CLASS('SYSOUT')/PRINT/PUNCH/HOLD/NIGHT/PRIVACY/LOWER
```

with the latter six mutually exclusive keywords available for the most frequently used 'SYSOUT' values A,B,H,N,P, and L.

The importance of the syntax chosen should not be underestimated.

(c) HDL has adopted simple character standards for dataset dispostion information in connection with a keyword:

A — Print at a remote station.
B — Punch at a remote station.
D — Allocate to dummy.
E — Print an edited dataset.
H — Hold in the output queue.
N — Make no allocation (has been preallocated).
P — Store in a private library, the name of which is generated from other syntax information.
T or * — Print at the terminal.
W — Store on a system-selected disk (usually a storage or work disk that is scratched at the end of the day).

Thus, if PRINT represents a keyword controlling the dispositon of a SYSPRINT dataset, PRINT(A) means print it at a remote station (which may also be specified by a DEST keyword). Not all the possibilities are permitted in every case, but consistent use is important to users.

(d) Avoid using Fortan files generally, if possible. Subroutines FASTIO, TSPOPN, and TSGOPN all have the means to read and write with nonstandard files. If an interface requires a

Fortran file (such as subroutines FSORT), choose one that does not interfere with any the user may have allocated. Subroutines DYNALC with verb code 7 may be used to check if a given ddname is allocated or not. In this way, a command need never unallocate a file that the user may have allocated for some other purposes.

(e) As mentioned earlier, let the system pick a ddname whenever possible to avoid user conflicts. IBM compilers, utilities, and other program products have default ddnames for needed files, but have provision to specify alternate names. If the system picks the ddname, specify the convertible attribute. If a specific ddname is required, specify the permanently allocated attribute without the convertible attribute (exactly how the TSO ALLOCATE command would do it). In this way, a system-selected ddname will begin with "SYS0", because the system will never pick an existing allocation that does not have the convertible attribute if you specify that it will have it. No conflicts can result if this is consistently used.

(f) Avoid unallocating a file at cleanup if its disposition does not conflict with other users' needs. This applies to all allocations, and is reasonable because most HDL logon procedures use DYNAMNBR=75 (i.e., 75 allocations with the permanently allocated attribute can exist at one time in addition to those specified by the logon procedure). In this way, an allocation may be reused, if needed, minimizing system overhead. The biggest problem arising from this procedure comes from users accessing a dataset in batch and TSO concurrently, using conflicting dispositions. If a dataset has been allocated with OLD, MOD, or NEW in the foreground, it cannot be allocated in the background. Thus, if a user creates a dataset with the ALLOCATE or CREATE commands and tries to write into it with a batch job without remembering to unallocate the dataset in TSO, the conflict may cause the operators to cancel the batch job.

Allocation of a ddname used at logon should be reallocated by a command because users may expect that allocation to be in force for subsequent work. For example, SYSPRINT is allocated to the terminal at logon for sophisticated users to run utilities. If changed by a command, such users may be taken by surprise when no output is printed at the terminal.

(g) Provide clear and accurate error messages. This will prevent angry frustration at not only having a command fail, but also getting a cryptic message. Some subroutines have built-in messages (ALCMSG, RECMSG, ALLC). It is important to try to anticipate, intercept, and explain possible user and system errors.

A common error source is allocation of a dataset with member for input. The allocation will succeed if the dataset exists but the member does not. An error will occur only when the attempt to open the dataset is made. Use subroutine ALLC to make necessary checks and provide resulting messages.

(h) Commands that do not require any cleanup and are not likely to have their return code checked for any reason by a user need not have an attention handler. If attention is hit in that case, the command ends immediately (and no return code is set). If an attention handler is used, check the attention flag often so that exit will be affected quickly after it is signalled.

5. EXAMPLES OF COMMANDS

Appendices B, C, and D contain three examples of commands to clarify some of the concepts described. All were written for IBM's H Extended compiler and are in use at HDL.

In appendix B, the SYSOUT command is listed, slightly modified for use in this report. It was primarily written to provide a COPIES keyword, sadly lacking in the IBM ALLOCATE command. Two other keywords also not available in the ALLOCATE command were added:

CLOSE—unallocates the file at close.

FORM—permits a form name to be specified.

All the allocation texts are initialized in the INTEGER*2 type statement. Note that the ddname is a required parameter; the call to set it comes first (a PARPOS call), with other keyword names defined next. COPY is defined as an alias to COPIES. HOLD and NOHOLD are mutually exclusive. Limits for values for those keywords having value can be found in the JCL manual. The USING keyword must be set to a ddname because the ATTRIBUTE command sets up a dummy file with fixed DCB attributes. Hence it is a ddname. The form name for the FORM keyword can have at most four characters, any of which can be alphameric or national. An integer value is entered for COPIES. The parsing call

```
CALL PARDEC(COP,1)
COP=1
```

initializes the number of copies to one if the user does not specify the keyword or specifies the keyword without value. Character data, however, must be intialized after the call to PARSE.

As it stands, an invalid number (say "A") entered would cause the prompt

```
INVALID NUMBER - REENTER
```

because "NUMBER" is the invalid message default. To get

```
INVALID NUMBER OF COPIES - REENTER
```

the proper call would be

```
CALL PARDEC(COP,'/NUMBER OF COPIES/',1)
```

To cause a prompt for the COPIES keyword when no value was supplied, we can use

```
CALL PARDEC(COP,1,1,'/NUMBER OF COPIES/')
```

or

```
CALL PARDEC(COP,'/NUMBER OF COPIES/',1,1,'/NUMBER OF COPIES/')
```

followed by the initializing statement for COP if the keyword is not used at all.

Note that special handling is required because COP must be a 4-byte integer in the call to PARDEC, but only the last byte is needed to stuff into the COPY(4) text. The method chosen was to equivalence COP with an INTEGER*2 array COPIES with dimension 2. Hence, COPIES(2) contains the last 2 bytes of COP. To move the binary value entered from byte 4 of COP to byte 3 of COP, the statement

```
COP=COP*256
```

is used. Finally, the statement

```
COPY(4)=COPIES(2)
```

stuffs the 8 binary bits correctly. Other methods could have been used. This technique is important for decimal values involving an odd number of bytes in the length (for primary and secondary space, directory blocks, etc.) as required by the allocation texts.

The rest of the command is straightforward, with heavy emphasis on good error messages. The REUSE keyword simply frees the ddname because it is unlikely the filename is associated with a sysout file.

Appendix C is a listing of the READN command, which uses TSPTGT in the print inhibit mode. It is intended to enable users to prompt for passwords in a Clist, and use the information within the Clist. The crucial routines are TSPTGT with IEC=3 (PUTGET facility, because GETLINE does not have a print inhibit mode), and SETCSV to set a Clist variable.

Before calling any parsing routines, subroutine GETCSV is used just to determine whether the user is in a Clist. An appropriate error message is sent if he is not. The first parsing call determines which Clist variable is to be set to the input. This is a required parameter. Since the PUTGET facility requires some string to write, the first call to PARSTR has a blank character as a default, so that LMSG(1) will never be zero. Thus, OPT=18 is specified to make at least one string required, but with a default.

Four other calls to PARSTR in a loop are for optional messages (IOPT=16 just defines the length LMSG to be INTEGER*4) the user may want to include a message sequence before the prompt response. Optional positional parameters are permitted for strings.

RETURN/NORETURN are the only optional keyword parameters.

If the parse is successful, another call is mode to GETCSV to check if the Clist name supplied by the user is in the Clist, and how many characters it has been set to. The SETCSV routine to be used later requires the variable to be set to have been defined in the Clist to at least the number of characters its new value is to have.

Next, TSPTGT is called to provide the messages and to receive an answer. IOP=94 if RETURN was specified or defaulted, or 126 if NORETURN was specified. The 94 includes

Value	Meaning
64	to sound a beep to alert the user that input is needed.
16	to inhibit printing the user's response.
8	to fold input to upper case.
4	to fold output to upper case (not needed because Fortran stores all alphabetic characters in upper case).
2	to request a multilevel message. The user can type "?" to get more information.
94	Total

Finally, the SETCSV routine is called to set the Clist variable. Clear and all-inclusive error messages are provided throughout.

Appendix D lists the COMPRESS command to illustrate the attention handler, alternate ddnames, various dynamic allocations, and the ATTACH routine to call the copy utility SPFCOPY.

After the parse, the attention handler is set up. Next a check is made on the entered dsname, because the user may have just entered a member name. 'SYS1.LPALIB' is rejected as a dsname. Subroutine RECFM is used to check the existence of the dataset name. The COMMON /ATTR/ is transmitted from RECFM so that the volume may be used in dynamic allocation to avoid another catalog search, and to check if the dsname was a PDS. After the various keys are adjusted depending on the keywords used, DYNALC allocates the dataset name, and its ddname is stuffed into the SYSIN control card positions at STR(12) and STR(27) with subroutine CHMOV. Note that if RELEASE is specified, the allocation is equivalent to

```
ALLOC DS( dsname') SPACE(1) TR MOD RLSE VOL('volser')
```

which releases space to the nearest track upon unallocation.

The VIO SYSIN file is next allocated with one track and the line

```
COPY INDD=ddname,OUTDD=ddname
```

is inserted into it with the TSPOPN, TSPPUT, and TSPCLS routines. This is an instance where the allocation for a system-generated ddname for the input PDS allocation is assumed, because an eight-character name is stuffed. The control card would be wrong if less than eight characters were returned by the system. The alternate SYSIN ddname is stuffed into the alternate ddname list.

Next, SYSUT3 and SYSUT4 are allocated and their ddnames are stuffed into the alternate ddname list. SYSPRINT is then allocated to DUMMY or a terminal file. Here both the convertible attribute and permanently allocated attribute are used, the former to prevent any existing allocation made on TSO (without the convertible attribute) from being selected, and the latter to prevent the system from automatically unallocating the file at the end of the command, so the allocation may be reused at some future time. The SYSPRINT ddname is stuffed into the alternate ddname list.

The only time the attention flag is checked is right before issuing a warning not to hit attention. This is primarily because all four allocations needed for SPFCOPY involve no dsname, and are quickly processed. The ATTACH subroutine is then called to execute SPFCOPY, functionally equivalent to IEBCOPY. A message is bypassed if attention was signalled, contrary to instructions, or SYSPRINT was printed at the terminal.

If the system or program abends, the abend code is transmitted by LC as follows: –LC will have the hex value of 01SSSUUU where SSS is the system abend code and UUU is the user abend code. These are easily extracted by the TSHEX routine. In this case, only the system code was extracted and a message sent in case of abend, because it was deemed unlikely that the SPFCOPY program would abend.

Finally, the compressed dataset is unallocated unless it had been allocated SHR in the first place.

6. CONCLUSIONS AND RECOMMENDATIONS

The routines and techniques developed in the last few years at HDL enable sophisticated commands to be written in Fortran for use in TSO. Overall systems efficiency is considerably improved when using commands instead of Clists, and at the same time the commands are quite easy to write, maintain, and debug.

However, the facility is not yet complete. There are three items which still should be developed:

(a) A facility to execute a Clist from a command.

(b) Error control for commands to enable cleanup operations if the command abends. Such a facility was provided in the ATTACH subroutine.

(c) PARSE should be reset before it exits to enable using it for subcommand parsing.

Appendix A.—Major Assembler Interface Subroutines

PROGRAM: ATTACH CATEGORY #: Z

CATEGORY: ALL OTHERS

TITLE:
ATTACH IS A FORTRAN-CALLABLE SUBROUTINE TO EXECUTE ANY PROGRAM.

TYPE: S SOURCE: AS DATE LAST CHG: 820503

AUTHOR: BILL JACOB, IBM (MODIFIED BY B. H. AUDET, 00251)
(MODIFIED BY W.C. BODYCOMB, IBM)

PURPOSE:
THIS ENABLES WRITING PROGRAMS THAT CALL OTHER PROGRAMS. IT IS AN ABSOLUTE NECESSITY FOR WRITING COMMANDS TO EXECUTE COMPILERS.

CALLING SEQUENCE:
CALL ATTACH(IOPT,IRC,PGM,PARM1,PARM2, ...)

ARGUMENTS:

NAME	TYPE	DIM	DESCRIPTION
IOPT	INT*4	NO	INPUT CONTROL OF ATTENTION INTERRUPTS. IF = 0, THIS PROGRAM DOES NO HANDLING OF ATTENTION INTERRUPTS. IF = 1, THE PROGRAM BEING ATTACHED IS STOPPED WHEN THE ATTENTION KEY IS HIT, UNLESS IT HAS ITS OWN ATTENTION HANDLER
IRC	INT*4	NO	THE OUTPUT RETURN CODE FROM THE PROGRAM BEING ATTACHED AND = 1 IF THE PROGRAM WAS INTERRUPTED BY ATTENTION. ANY OTHER NEGATIVE VALUE MEANS THE PROGRAM BEING ATTACHED ABENDED. -IRC WILL HAVE THE HEX VALUE OF 01SSSUUU, WHERE SSS IS THE SYSTEM ABEND CODE AND UUU IS THE USER ABEND CODE.
PGM	8 BYTES OR 16 BYTES		THE NAME OF THE PROGRAM BEING ATTACHED, PADDED WITH BLANKS TO 8 CHARACTERS. IT MUST BE A NAME IN THE LINK PACK AREA OR A MEMBER IN ONE OF THE LINK LIST LIBRARIES. HOWEVER, IF THE NAME BEGINS WITH "SYS0", IT IS ASSUMED TO BE A DDNAME TO WHICH A LOAD LIBRARY IS ALLOCATED. THE NEXT 8 BYTES MUST THEN BE THE NAME OF THE MEMBER TO BE EXECUTED. IF THE DDNAME IS SUPPLIED, THE LIBRARY IS USED AS A TASKS LIBRARY SO THAT SUBSEQUENT LOADS OR ATTACHES PERFORMED BY THE ATTACHED PROGRAM WILL BE RESOLVED TO THIS LIBRARY.
PARM1	ARRAY	YES	THIS AND ANY OTHER PARAMETERS ARE OPTIONAL, AND IF PRESENT, WILL BE PASSED TO THE PROGRAM BEING ATTACHED.

RESTRICTIONS:
NONE

REFERENCES/ACKNOWLEDGEMENTS:
NONE

APPENDIX A

PROGRAM: ATTEN CATEGORY #: Z

CATEGORY: ALL OTHERS

TITLE:
TSO ATTENTION INTERCEPT ROUTINE.

TYPE: S SOURCE: AS DATE LAST CHG: 821015

AUTHOR: BILL JACOB

PURPOSE:
ATTEN INTERCEPTS AN ATTENTION INTERRUPT AND SET ITS ARGUMENT TO A NONZERO VALUE. AN ENTRY CALLED ATTOFF DISABLES THE INTERCEPT.

CALLING SEQUENCE:
CALL ATTEN(I)
CALL ATTOFF

ARGUMENTS:

NAME	TYPE	DIM	DESCRIPTION
I	INT*4 OR LOG*4 OR LOG*1	NO	OUTPUT ATTENTION FLAG THAT IS SET TO A NONZERO VALUE (X'01') WHEN THE TERMINAL USER CAUSES AN INTERRUPT. IT SHOULD BE SET TO ZERO (OR .FALSE.) BEFORE ENTRY TO THE ROUTINE. IT MAY BE RESET TO ZERO (OR .FALSE.) FOLLOWING A CHECK ON ON ITS VALUE THAT RESULTED IN A NONZERO (OR .TRUE.) VALUE, OR AFTER ATTOFF IS CALLED. (A DIFFERENT ARGUMENT MAY BE USED AFTER A CALL TO ATTOFF).

RESTRICTIONS:
(1) ATTEN IS CALLED AT THE BEGINNING OF THE PROGRAM, AND MAY BE CALLED AGAIN AFTER IT IS DISABLED WITH ATTOFF. WHEN THE ATTENTION INTERCEPT IS ENABLED, THE USER PERIODICALLY CHECKS THE VALUE OF THE ATTENTION FLAG, I, TO SEE IF IT HAS A NONZERO VALUE. IF IT DOES, APPROPRIATE ACTION SHOULD BE TAKEN (CLEANUP AND EXIT, RESTART AT A SPECIFIC POINT IN THE PROGRAM, ETC.) THIS IS PARTICULARLY IMPORTANT FOLLOWING I/O OPERATIONS OR AFTER PROGRAM LOOPS. IF PROGRAM CONCLUSION IS NOT PART OF THE ACTION TAKEN, THE ATTENTION FLAG, I, SHOULD BE RESET TO ZERO, OR THE PROGRAM WILL THINK AN INTERRUPT OCCURRED THE NEXT TIME I IS CHECKED.
(2) THIS ROUTINE DOES NOT PREVENT THE PROGRAM LOSING CONTROL IF TWO INTERUPTS OCCUR TOGETHER. WITHIN A SINGLE LOAD MODULE (WITH ANY ONE COPY OF ATTEN), ONLY ONE ATTENTION EXIT WILL BE ACTIVE.
(3) IF OPT(2) IS USED IN A FORTRAN PROGRAM, IT IS WISE TO PUT THE ATTENTION FLAG, I, IN COMMON, TO PREVENT SHIFTING OF STATEMENTS CONTAINING I BY THE OPTIMIZER.

REFERENCES/ACKNOWLEDGEMENTS:
NONE

PROGRAM: COMAND CATEGORY #: Z

CATEGORY: ALL OTHERS

TITLE:
EXECUTES A TSO COMMAND FROM ANOTHER COMMAND.

TYPE: S SOURCE: AS DATE LAST CHG: 800805

AUTHOR: BILL JACOB, IBM; MODIFIED BY WALTER C. BODYCOMB, IBM

PURPOSE:
WHEN WRITING OTHER COMMANDS, IT IS OFTEN NECESSARY TO PERFORM A TASK FOR WHICH NO PRIMARY SUBROUTINE HAS BEEN WRITTEN (E.G., PROTECT). THIS SUBROUTINE ALLOWS THE USE OF ANY IBM OR HDL TSO COMMAND. PROVISION IS ALSO MADE FOR TRANSMITTING BACK CHARACTER INFORMATION FROM THE CALLED COMMAND, IF THAT COMMAND WAS CODED TO SUPPLY IT.

CALLING SEQUENCE:
CALL COMAND(IOPT,IRC,CMDNAM (,RTNLEN,RTNARG)(,CMDRGS ...))

ARGUMENTS:

NAME	TYPE	DIM	DESCRIPTION
IOPT	INT	NO	ATTENTION HANDLER ARGUMENT (INPUT). SET TO 0, IGNORE ATTENTION AND NO 'RTNARG' IS EXPECTED. (TSO RETURNS TO "READY" MODE FOR ATTENTION.) 1, WILL STOP 'CMDNAM' IF THE ATTENTION KEY IS HIT, AND 'RTNARG' IS NOT EXPECTED. 2, IGNORE ATTENTION AND 'RTNARG' IS EXPECTED. (TSO RETURNS TO "READY" MODE FOR ATTENTION.) 3, WILL STOP 'CMDNAM' IF THE ATTENTION KEY IS HIT, AND 'RTNARG' IS EXPECTED.
IRC	INT	NO	OUTPUT RETURN CODE: -1 MEANS THE USER SIGNALLED ATTENTION, -2 MEANS THE SUBROUTINE IS NOT BEING CALLED FROM A TSO COMMAND, -3 MEANS THERE IS A SYNTAX ERROR IN THE CALLING SEQUENCE OR ARGUMENTS FOR THIS SUBROUTINE, ANY OTHER VALUE IS THE RETURN CODE OF 'CMDNAM'.
CMDNAM	CHAR 8 BYTES		THE NAME OF THE TSO COMMAND TO BE EXECUTED (INPUT). IT SHOULD BE LEFT JUSTIFIED AND PADDED WITH BLANKS TO 8 CHARACTERS.
RTNLEN	INT*4	NO	INPUT/OUTPUT VARIABLE. AS INPUT, IT IS THE MAXIMUM NUMBER OF BYTES EXPECTED FOR 'RTNARG'. AS OUTPUT, IT IS THE NUMBER OF BYTES RETURNED TO 'RTNARG'. THE OUTPUT VALUE WILL BE LESS THAN OR EQUAL TO THE INPUT VALUE.
RTNARG	CHAR AS REQ'D		THE AREA IN WHICH THE STRING RETURNED FROM 'CMDNAM' IS PLACED. PROVISION MUST BE MADE TO SUPPLY THE STRING FROM 'CMDNAM', AS IT HAS BEEN IN SUBROUTINE "QUIT".

APPENDIX A

PROGRAM: COMAND CATEGORY #: Z

ARGUMENTS:

NAME	TYPE	DIM	DESCRIPTION
CMDRGS	CHAR	YES	ONE OR MORE ARGUMENTS THAT SUPPLY THE COMMAND KEY-WORDS (INPUT). TWO FORMATS ARE ACCEPTABLE AND THEY MAY BE INTERMIXED AS NECESSARY TO SUPPLY ALL THE KEYWORDS: 1) A SELF-DEFINING CHARACTER STRING. THE FIRST CHARACTER IS TAKEN TO BE A DELIMITER AND THE END OF THE STRING IS THE NEXT OCCURANCE OF THIS SAME DELIMITER. IN THIS FORM, A COUNT NEED NOT BE GIVEN. (THE DELIMITER SHOULD BE SOME PRINTABLE CHARACTER.) THE STRING IS ADDED TO THE COMMAND KEYWORD STRING. 2) A DOUBLE ARGUMENT CHARACTER STRING. THE FIRST IS AN INTEGER GIVING THE NUMBER OF CHARACTERS IN THE STRING. THE SECOND IS THE STRING THAT IS TO BE ADDED TO THE COMMAND KEYWORD STRING. ONE BLANK IS AUTOMATICALLY INSERTED BETWEEN THE COMMAND NAME AND ANY ARGUMENTS (CMDRGS) SUPPLIED BY THE USER.
	OR		
	INT	NO	

RESTRICTIONS:
SOME COMMANDS MAY NOT WORK, BUT TESTING IS REQUIRED TO DETERMINE THIS.
AS AN EXAMPLE OF THE USE OF THIS ROUTINE, THE CALL TO USE THE LIST COMMAND ON THE DATASET 'SYS1.PAGSRCE(COMAND)' WITH NONUM FOLLOWS. IT ASSUMES THE NAME OF THE MEMBER (COMAND) IS IN A REAL*8 VARIABLE CALLED MEM WITH LMEM AS ITS LENGTH:

```
CALL COMAND(1,IRC,'LIST     ','/''SYS1.PAGSRCE(/',LMEM,MEM,'/''' NONUM/')
```

BOTH FORMS OF CMDRGS ARE USED WITH A SLASH (/) AS THE SELF DEFINING DELIMITER. NOTE THAT THE QUOTE NEEDED IN THE COMMAND SYNTAX IS DOUBLED UP INSIDE THE QUOTED STRING ARGUMENT.

THE SYSTEM LOOKS FOR THE COMMAND IN THE USUAL PLACES (THE LINK PACK AREA AND THE LINK LIST DATASETS).

THE ARGUMENTS 'RTNLEN' AND 'RTNARG' ARE MEANINGFUL ONLY IF PROVISION WAS MADE TO SUPPLY ALPHA INFORMATION BACK TO THE CALLING COMMAND AS WITH SUBROUTINE "QUIT".

REFERENCES/ACKNOWLEDGEMENTS:
NONE

PROGRAM: DYNALC CATEGORY #: Z

CATEGORY: ALL OTHERS

TITLE:
DYNAMIC ALLOCATION CALL INTERFACE

TYPE: S SOURCE: AS DATE LAST CHG: 820511

AUTHOR: BILL JACOB

PURPOSE:
THE PROGRAM WILL DYNAMICALLY ALLOCATE DATA SETS DURING THE RUNNING OF A PROGRAM. THIS INCLUDES ALLOCATE, CREATE, FREE, AND DELETE.

CALLING SEQUENCE:
CALL DYNALC(IVERB,IRETCD,IERRCD,INFOCD,NTEXTS,T1,T29,...)

ARGUMENTS:

NAME	TYPE	DIM	DESCRIPTION
IVERB	I*4		DYNAMIC ALLOCATION VERB CODE. THIS VALUE SHOULD BE SUPPLIED BY THE USER TO INDICATE WHICH OF THE MAJOR DYNAMIC ALLOCATION FUNCTIONS HE IS REQUESTING. (SEE THE MANUAL REFERENCED BELOW.) HOWEVER, IVERB=8 CAN BE USED WITH NO OTHER ARGUMENTS TO FORCE THE NEXT DYNAMIC ALLOCATION TO NOT USE AN EXISTING ALLOCATION. IT IS RECOMMENDED THAT CALL DYNALC(8) PRECEDE A CALL TO ALLOCATE A DUMMY FILE IF THAT ALLOCATION IS MADE WITHOUT THE CONVERTIBLE ATTRIBUTE.
IRETCD	I*4		DYNAMIC ALLOCATION RETURN CODE. THIS VALUE WILL BE SET BY 'DYNALC' TO THE RETURN CODE IT RECEIVES FROM THE SYSTEM DYNAMIC ALLOCATION ROUTINE.
IERRCD	I*4		ERROR CODE FOR DYNAMIC ALLOCATION. THIS VALUE WILL BE SET BY 'DYNALC' TO THE DYNAMIC ALLOCATION ERROR CODE RETURNED BY THE SYSTEM. IT IS OF INTEREST, GENERALLY, ONLY WHEN 'IRETCD' IS NONZERO.
INFOCD	I*4		DYNAMIC ALLOCATION INFORMATIONAL CODE. THIS VALUE WILL BE SET BY 'DYNALC' TO THE INFORMATIONAL REASON CODE RETURNED BY THE SYSTEM. IT IS OF INTEREST, GENERALLY, ONLY WHEN 'IRETCD' IS NONZERO.
NTEXTS	I*4		NUMBER OF TEXT UNITS TO BE USED. THIS PARAMETER IS USED BY THE CALLER TO TELL 'DYNALC' HOW MANY OF THE FOLLOWING TEXT UNIT ARGUMENTS ARE TO BE PASSED TO THE SYSTEM DYNAMIC ALLOCATION ROUTINE. IF 'NTEXTS' IS ZERO, ALL OF THE TEXT UNIT ARGUMENTS CODED IN THE CALLING STATEMENT WILL BE PASSED TO THE SYSTEM. WHEN NON-ZERO (MUST BE POSITIVE), 'NTEXTS' CAN BE USED BY THE PROGRAMMER TO CONDITIONALLY PASS TEXT UNITS: ONLY THE FIRST 'NTEXTS' TEXT UNITS WILL BE PASSED TO THE SYSTEM ROUTINE.

APPENDIX A

PROGRAM: DYNALC CATEGORY #: Z

ARGUMENTS: CONTINUED

T1-TN CHAR YES DYNAMIC ALLOCATION TEXT UNITS. EACH TEXT UNIT OCCUPIES ONE PARAMETER IN THE SUBROUTINE CALLING STATEMENT FOR 'DYNALC'. IN GENERAL, A TEXT UNIT CORRESPONDS TO A JCL KEYWORD AND IT'S VALUE OR TO A KEYWORD ON THE TSO 'ALLOCATE' OR 'ATTRIB' COMMANDS. A TYPICAL TEXT UNIT CONTAINS:
- THE TEXT UNIT KEY INDICATING WHAT KIND OF INFORMATION IS CONTAINED IN THIS TEXT UNIT. IF 0, IT WILL BE IGNORED.
- A NUMBER INDICATING HOW MAY VALUES ARE SUPPLIED IN THIS TEXT UNIT (GENERALLY ONE).
- A NUMBER INDICATING HOW LONG THE FIRST VALUE IS.
- THE VALUE BEING SUPPLIED.

THESE LAST TWO ELEMENTS MAY APPEAR MORE THAN ONCE IF THE SECOND ELEMENT INDICATES THAT SEVERAL VALUES ARE BEING SUPPLIED. SOME TEXT UNITS REQUIRE ONLY THE TEXT UNIT KEY AND NO OTHER INFORMATION. THE FIRST THREE ELEMENTS IN EACH TEXT UNIT ARE HALFWORD INTEGERS (INTEGER*2). THE LAST ELEMENT WILL VARY IN LENGTH ACCORDING TO THE TYPE OF TEXT UNIT AND THE INFORMATION BEING SUPPLIED. FOR EXAMPLE, THE TEXT UNIT FOR A DATASET NAME OF 'SYS1.PAGLOAD' MIGHT BE SUPPLIED BY:

```
      INTEGER*2 DSN(9)/2,1,12,'SY','S1','.P','AG',
     X   'LO','AD'/
      CALL DYNALC( ..... ,DSN, ... )
```

THE TEXT UNIT KEY IS 2; 1 VALUE IS BEING SUPPLIED. THE VALUE IS 12 CHARACTERS LONG, AND IS SUPPLIED AS SHOWN ABOVE. CONSTRUCTION OF THESE TEXT UNITS IS NOT POSSIBLE WITHOUT RECOURSE TO THE MANUAL CITED BELOW. NOTE THAT, IN THE MANUAL, SOME INFORMATION SUCH AS THE TEXT UNIT KEYS IS GIVEN IN HEXADECIMAL. THE USER SHOULD TAKE CARE TO CONVERT THESE TO DECIMAL AS REQUIRED.

RESTRICTIONS:
THIS PROGRAM SHOULD NOT BE USED BY THE CASUAL PROGRAMMER. HIGHER LEVEL FORTRAN CALLABLE ROUTINES WILL BE WRITTEN THAT MAKE USE OF THIS SUBROUTINE, AND WILL BE USED FOR SPECIFIC OPERATIONS SUCH AS FILE ALLOCATION.
A NONCALLABLE ENTRY POINT, DYNALB, IS AVAILABLE FOR DAIRFAIL PROCESSING USING SUBROUTINES DYNALE, DYNALW, OR DYNALM.

REFERENCES/ACKNOWLEDGEMENTS:
THIS SUBROUTINE REFERS TO THE SYSTEM LIBRARY MANUAL THAT DESCRIBES THE ALLOCATION PARAMETERS.
THE MANUAL IS CALLED OS/VS2 SYSTEM PROGRAMMING LIBRARY: JOB MANAGEMENT - GC28-0627.

PROGRAM: PARSE CATEGORY #: Z

CATEGORY: ALL OTHERS

TITLE:
A CALL INTERFACE TO THE TSO PARSE ROUTINE.

TYPE: S SOURCE: AS DATE LAST CHG: 780725

AUTHOR: BILL JACOB, IBM

PURPOSE:
WILL ALLOW THE WRITER OF TSO COMMANDS TO FULLY UTILIZE THE TSO PARSER, INCLUDING PROMPTS, ERROR DETECTION, RECOGNITION OF MUTUAL EXCLUSIVE KEYWORDS, DATASET NAMES WITH PASSWORDS, KEYWORD PARAMETER ABBREVIATIONS, ETC.

CALLING SEQUENCE:
THERE ARE 8 ENTRY POINTS:

```
CALL PARDSN(DSN,MEM,PSWD,LENS,LIST(,NUM),IOPT(,PRODEF))
CALL PARPOS(DATA,LENS,TYPE,LIST(,NUM),IOPT,MAXL,IFIRST,IOTHER(,PRODEF))
CALL PARDEC(INTGS(,TYPE),LIST(,NUM)(,IOPT,PRODEF))
CALL PARSTR(STR,LENS,MAXL(,IOPT(,PRODEF)))
CALL PARKEY(IWHICH(,IDEFLT))
CALL PARNAM(NAME(,ISUBF)(,-1,INSERT)(,ALIAS1)(,ALIAS2)...)
CALL PARSUB(ISUBF)
CALL PARSE(IRETCD(,CMD))
```

THE ORDER OF CALL IS SOMETIMES IMPORTANT. PARENTHESIZED VARIABLES ARE EITHER OPTIONAL OR MUST SOMETIMES BE OMITTED. SEE RESTRICTIONS.

ARGUMENTS: (FOR ENTRY PARDSN)

NAME	TYPE	DIM	DESCRIPTION
DSN	44 BYTES	*LIST	OUTPUT VARIABLE SET BY PARSE TO THE DATASET NAMES ENTERED, WITHOUT MEMBER OR PASSWORD; PADDED WITH BLANKS TO 44 BYTES FOR EACH NAME. IF LIST=5, APPROPRIATE DIMENSION STATEMENTS ARE INTEGER*4 DSN(11,5) OR INTEGER*2(22,5)
MEM	8 BYTES	*LIST	SET TO THE MEMBER NAMES ENTERED, PADDED WITH BLANKS. TYPICAL DIMENSION STATEMENTS: REAL*8 MEM (LIST=1); INTEGER MEM(2) (LIST=1); REAL*8 MEM(5) (LIST=5); INTEGER*2 MEM(4,5) (LIST=5)
PSWD	8 BYTES	*LIST	SET TO THE PASSWORDS ENTERED, PADDED WITH BLANKS. TYPICAL DIMENSION STATEMENTS ARE SIMILAR TO MEM.
LENS	I*2 OR I*4	3*LIST	AN INTEGER ARRAY SET TO THE LENGTHS OF DSN, MEM, AND PSWD. IF ANY ARE OMITTED, LENS IS SET TO 0. IOPT DETERMINES IF LENS IS A 2-BYTE OR 4-BYTE INTEGER. TYPICAL DIMENSIONS ARE SIMILAR TO MEM: INTEGER*2 LENS(3,5) (LIST=5), ETC. IF DSN IS ENTERED IN QUOTES (APOSTROPHES), THE APPROPRIATE ELEMENT OF LENS WILL BE NEGATIVE.
LIST	I*4	NO	THE NUMBER OF DSNAMES ALLOWED (INPUT VARIABLE). IF LIST=1, THEN NUM MUST BE OMITTED.
NUM	I*4	NO	THE NUMBER OF DSNAMES ENTERED (0-255). MUST BE OMITTED IF LIST=1. THIS IS AN OUTPUT VARIABLE.
IOPT	I*4	NO	A SET OF OPTIONS WHICH CAN BE SELECTED BY ADDING

APPENDIX A

PROGRAM: PARSE CATEGORY #: Z

ARGUMENTS: CONTINUED

THE FOLLOWING DECIMAL VALUES:
- 0 - PARAMETER NOT REQUIRED.
- 1 - THIS DSNAME IS REQUIRED; 'PRODEF' IS A PROMPTING MESSAGE (AS IN "ENTER 'PRODEF'")
- 2 - THIS DSNAME IS REQUIRED; 'PRODEF' IS A DEFAULT VALUE.
- 4 - AN ASTERISK IS ACCEPTABLE AS THE DSNAME, OR AS ANY ONE LEVEL OF THE DSNAME, E.G., 'HA25XX.*.FORT'.
- 8 - THE TSO USER PREFIX IS **NOT** TO BE LEFT-APPENDED TO THE DSNAME ENTERED, IF UNQUOTED.
- 16 - LENS IS AN I*4 VARIABLE RATHER THAN THE DEFAULT I*2 VARIABLE.

PRODEF AD YES THE PROMPTING PHRASE OR DEFAULT PHRASE, DEPENDING ON IOPT. IF NEITHER 1 OR 2 IS SPECIFIED IN IOPT, THEN PRODEF IS NOT REQUIRED AND IS IGNORED IF SUPPLIED. IN THAT CASE, LENS WILL CONTAIN ZEROES IF THE USER DOES NOT ENTER A DSNAME.

NOTE ON AD: THE TYPE AD IS USED HERE AND BELOW TO INDICATE THAT THE PARAMETER IS A CHARACTER STRING WHICH IS TO BE SURROUNDED BY SELF-DEFINING DELIMITERS (AS IN THE TSO FIND AND CHANGE COMMANDS). THE FINAL DELIMITER MAY NOT APPEAR WITHIN THE CHARACTER STRING. FOR EXAMPLE:

```
CALL PARDSN(DSN,...,1,'/DATASET TO BE COMPILED/')
```

THE 1 INDICATES THE DATASET IS REQUIRED; IF OMITTED, THE PROMPTING MESSAGE "ENTER POSITIONAL PARAMETER DATASET TO BE COMPILED" IS DISPLAYED

ARGUMENTS: (FOR ENTRY PARPOS)

NAME	TYPE	DIM	DESCRIPTION
DATA	ANY	MAXL*LIST	OUTPUT VARIABLE SET BY PARSE TO THE SET OF PARAMETERS ENTERED BY THE USER; PADDED BY BLANKS TO THE VALUE OF MAXL.
LENS	I*4 OR I*2	LIST	THE LENGTH OF EACH 'DATA' (OUTPUT VARIABLE). IT IS SET BY PARSE IN THE RANGE OF 0 (IF PARAMETERS NOT ENTERED) TO MAXL. IOPT DETERMINES IF LENS IS AN I*2 OR I*4 VARIABLE.
TYPE	AD	YES	SELF-DELIMITED STRING WHICH IS USED TO PROMPT THE USER IF THE ENTRY WAS INVALID, AS DEFINED BY IFIRST, IOTHER, AND MAXL. THE DISPLAYED STRING IS "INVALID 'TYPE', REENTER" THIS IS AN INPUT VARIABLE.
LIST	I*4	NO	THE NUMBER OF DATA NAMES ALLOWED (INPUT VARIABLE). IF LIST=1, THEN NUM MUST BE OMITTED.
NUM	I*4	NO	THE NUMBER OF DATA NAMES ENTERED (0-255). MUST BE OMITTED IF LIST=1.
IOPT	I*4	NO	A SET OF OPTIONS WHICH CAN BE SELECTED BY ADDING THE FOLLOWING DECIMAL VALUES: 0 - PARAMETER NOT REQUIRED; NO PROMPTING OR DEFAULTS AND PRODEF IS IGNORED IF SUPPLIED. 1 - THIS PARAMETER IS REQUIRED; 'PRODEF' IS A PROMPTING MESSAGE (AS IN "ENTER 'PRODEF'"). 2 - 'PRODEF' IS A DEFAULT VALUE FOR THIS PARAMETER.

PROGRAM: PARSE CATEGORY #: Z

ARGUMENTS: CONTINUED

APPLICABLE ONLY IF THIS PARAMETER IS REQUIRED.
4 - AN ASTERISK IS ACCEPTABLE FOR THIS PARAMETER.
8 - THIS PARAMETER IS NOT TO BE TRANSLATED TO UPPER CASE BY PARSE.
16 - LENS IS AN I*4 VARIABLE RATHER THAN THE DEFAULT I*2 VARIABLE.

MAXL	I*4	NO	THE MAXIMUM NUMBER OF BYTES OF 'DATA' (THE MAXIMUM VALUE OF ANY 'LENS'). THIS IS AN INPUT VARIABLE.
IFIRST	I*4	NO	RESTRICTION ON THE FIRST CHARACTER OF THIS PARAMETER 0 - ANY CHARACTER ACCEPTABLE. 1 - ALPHABETIC OR NATIONAL CHARACTER ONLY. 2 - NUMERIC ONLY. 3 - ALPHABETIC, NATIONAL, OR NUMERIC ONLY. 4 - ALPHABETIC ONLY. 5 - ALPHABETIC OR NUMERIC ONLY.
IOTHER	I*4	NO	RESTRICTION ON OTHER THAN THE FIRST CHARACTERS OF THIS PARAMETER (INPUT VARIABLE). THE SAME CODE AS 'IFIRST' IS USED.
PRODEF	AD	YES	THE PROMPTING STRING OR DEFAULT STRING (IF THIS IS A REQUIRED PARAMETER). 'PRODEF' IS IGNORED IF THIS IS NOT A REQUIRED PARAMETER. IN THAT CASE, LENS IS SET TO ZEROES. THIS IS AN INPUT PARAMETER.

ARGUMENTS: (FOR ENTRY PARDEC)

NAME	TYPE	DIM	DESCRIPTION
INTGS	I*4	LIST	OUTPUT VARIABLE SET BY PARSE TO THE FIXED POINT NUMBERS ENTERED BY THE USER.
TYPE	AD	YES	SELF-DELIMITED STRING WHICH IS USED TO PROMPT THE USER IF THE ENTRY WAS INVALID, AS DEFINED BY IFIRST, IOTHER, AND MAXL. THE DISPLAYED STRING IS "INVALID 'TYPE', REENTER" THIS IS AN OPTIONAL INPUT VARIABLE. IF OMITTED, ITS VALUE DEFAULTS TO '/NUMBER/'.
LIST	I*4	NO	THE NUMBER OF INTGS ALLOWED (INPUT VARIABLE). IF LIST=1, THEN NUM MUST BE OMITTED.
NUM	I*4	NO	THE NUMBER OF INTEGERS ENTERED (0-255). MUST BE OMITTED IF LIST=1. THIS IS AN OUTPUT VARIABLE.
IOPT	I*4	NO	A SET OF OPTIONS WHICH CAN BE SELECTED: 1 - 'PRODEF' IS A PROMPT STRING. THIS PARAMETER IS REQUIRED. 2 - 'PRODEF' IS A DEFAULT STRING (IN CHARACTER FORM). THIS PARAMETER IS REQUIRED. IF THIS PARAMETER IS NOT REQUIRED, IOPT AND PRODEF SHOULD BE OMITTED.
PRODEF	AD	YES	THE PROMPTING STRING OR DEFAULT STRING (IF THIS IS A REQUIRED PARAMETER). 'PRODEF' IS IGNORED IF THIS IS NOT A REQUIRED PARAMETER, AND IOPT WAS SPECIFIED.

ARGUMENTS: (FOR ENTRY PARSTR)

NAME	TYPE	DIM	DESCRIPTION
STR	ANY	YES	THE OUTPUT VARIABLE SET TO THE CHARACTER STRING THAT THE COMMAND LINE SPECIFIED.
LENS	I*2 OR I*4	NO	AN OUTPUT VARIABLE SET TO THE LENGTH OF 'STR'. THE TYPE DEPENDS ON 'IOPT'.

APPENDIX A

PROGRAM: PARSE CATEGORY #: Z

ARGUMENTS: CONTINUED

MAXL	I*4	NO	AN OUTPUT VARIABLE SET TO THE MAXIMUM VALUE OF 'LENS'.
IOPT	I*4	NO	A SET OF OPTIONS WHICH CAN BE SELECTED BY ADDING THE FOLLOWING DECIMAL VALUES. 0 - PARAMETER NOT REQUIRED; NO PROMPTING OR DEFAULTS AND 'PRODEF' IS IGNORED IF SUPPLIED. 1 - THIS PARAMETER IS REQUIRED; 'PRODEF' IS A PROMPTING MESSAGE (AS IN "ENTER 'PRODEF'"). 2 - 'PRODEF' IS A DEFAULT VALUE FOR THIS PARAMETER. APPLICABLE ONLY IF THIS PARAMETER IS REQUIRED. 8 - THIS PARAMETER IS NOT TO BE TRANSLATED TO UPPER CASE BY PARSE. 16 - LENS IS AN I*4 VARIABLE RATHER THAN THE DEFAULT I*2 VARIABLE.
PRODEF	AD	YES	THE PROMPTING STRING OR DEFAULT STRING (IF THIS IS A REQUIRED PARAMETER). 'PRODEF' IS IGNORED IF THIS IS NOT A REQUIRED PARAMETER. IN THAT CASE, LENS IS SET TO ZERO. THIS IS AN INPUT PARAMETER.

ARGUMENTS: (FOR ENTRY PARKEY)

NAME	TYPE	DIM	DESCRIPTION
IWHICH	I*4	NO	OUTPUT VARIABLE SET BY PARSE TO THE NUMBER OF THE PARNAM NAME WHICH THE USER ENTERED; ZERO IF NO DEFAULT AND NO KEYWORD FROM THE PARNAM LIST (DEFINED IN CALLS TO PARNAM FOLLOWING) WAS ENTERED.
IDEFLT	AD	YES	THE DEFAULT VALUE OF THE KEYWORD. MUST MATCH A PARNAM NAME OR ALIAS WHICH FOLLOW IN CALLS TO PARNAM. OMIT IF THERE IS NO DEFAULT.

ARGUMENTS: (FOR ENTRY PARNAM)

NAME	TYPE	DIM	DESCRIPTION
NAME	AD	YES	ONE OF THE POSSIBLE NAMES FOR THE KEYWORD (INPUT VARIABLE).
ISUBF	I*4	NO	INPUT NUMBER ONLY IF 'NAME' HAS A SUBFIELD (KEYWORD WITH VALUE). THE NUMBER IS ARBITRARY BUT MUST MATCH 'ISUBF' IN THE CALL TO PARSUB, AND MUST BE DISTINCT FROM ALL OTHER 'ISUBF' VALUES USED BEFORE THE CALL TO PARSE.
INSERT	AD	YES	KEYWORD(S) WHICH WILL BE LOGICALLY INSERTED INTO THE COMMAND BUFFER BY PARSE IF THE KEYWORD 'NAME' IS FOUND. USED IF ONE KEYWORD IMPLIES ANOTHER. IF INSERT IS USED, IT MUST BE PRECEDED BY -1. THIS IS AN INPUT VARIABLE.
ALIAS1	AD	YES	ALIAS FOR THE KEYWORD IN NAME, OTHER THAN THE STANDARD TSO ABBREVIATION CONVENTIONS. THIS IS AN INPUT VARIABLE.
ALIAS2	AD	YES	AS IN ALIAS1. OTHER ALIASES CAN BE DEFINED.

ARGUMENTS: (FOR ENTRY PARSUB)

NAME	TYPE	DIM	DESCRIPTION
ISUBF	I*4	NO	INPUT SUBFIELD IDENTIFICATION, WHICH MUST MATCH EXACTLY THE ISUBF PARAMETER OF ONE AND ONLY ONE PRECEDING PARNAM SUBROUTINE CALL. THIS IS AN INPUT PARAMETER.

```
PROGRAM:     PARSE              CATEGORY #: Z

ARGUMENTS:                          CONTINUED

 ARGUMENTS: (FOR ENTRY PARSE)
 NAME    TYPE    DIM DESCRIPTION
IRETCD    I*4    NO  OUTPUT RETURN CODE OF THE PARSE OPERATION:
                       0 - PARSE WAS SUCCESSFUL.
                       1 - PARSE WAS CALLED BEFORE ANY DESCRIPTIVE
                           SUBROUTINES WERE.
                       2 - ERROR IN PARSE'S SEARCH OF THE SAVE CHAIN AREA.
                       3 - PARSE WAS USED IN A BATCH OR TSO CALL
                           ENVIRONMENT.
                       4 - COMMAND PARAMETERS WERE INCOMPLETE, AND PARSE
                           WAS UNABLE TO PROMPT.
                       8 - PARSE WAS INTERRUPTED BY ATTENTION.
                      12 - INTERNAL ERROR IN ONE OF THESE SUBROUTINES.
                      16 - INSUFFICIENT MAIN STORAGE AVAILABLE.
                      20 - (SHOULD NOT BE RECEIVED.)
                      24 - INTERNAL ERROR IN ONE OF THESE SUBROUTINES.
                      28 - TERMINAL HAS BEEN DISCONNECTED.
                      NOTE: IF IRETCD IS ANYTHING OTHER THAN ZERO THEN
                      NONE OF THE USER VARIABLES SPECIFIED IN CALLS TO THE
                      OTHER SIX ENTRY POINTS CONTAINS ANY VALID INFORMA-
                      TION
 CMD   REAL*8    NO  COMMAND NAME ENTERED BY THE USER (OUTPUT VARIABLE);
                     PADDED WITH BLANKS TO LENGTH OF 8. 'CMD' IS SUPPLIED
                     EVEN IF 'IRETCD' = 1.

RESTRICTIONS:
USEABLE ONLY IN A TSO COMMAND ENVIRONMENT.
ALL OUTPUT VARIABLES ARE SET ONLY AFTER PARSE IS CALLED. THE ORDER OF
CALLS ARE AS FOLLOWS:
 (1)  PARDSN, PARDEC, OR PARPOS ARE CALLED FOR EACH POSITIONAL
      PARAMETER OF THE COMMAND.
 (2)  PARKEY FOLLOWED BY ONE OR MORE PARNAM FOR EACH KEYWORD.
 (3)  PARSUB FOLLOWED BY EITHER PARDSN, PARDEC, OR PARPOS FOR ALL
      KEYWORDS WITH VALUES. (PARKEY/PARNAM CAN FOLLOW PARSUB IF THE
      IF THE KEYWORD VALUE IS ANOTHER KEYWORD WITHOUT VALUE.)
 (4)  PARSE
*******************************************************************************
EXAMPLES -
(1) COMMAND:  AAAA 'DSNAME' 'VOLSER'      (DSNAME REQUIRED, VOLSER NOT)
        INTEGER*2 DSN(22),VOL(3),MEM(4),PAS(4),LEN(3),LVOL
        CALL PARDSN(DSN,MEM,PASS,LEN,1,1,'/NAME OF DATASET./')
        CALL PARPOS(VOL,LVOL,'/VOLUME SERIAL NUMBER/',1,0,6,3,3)
        CALL PARSE(IRC)
    NOTES: ONLY POSITIONAL PARAMETERS; HENCE NO PARKEY, PARSUB, OR
        PARNAM.
            1,1 IN PARDSN MEANS ONE DSNAME IS REQUIRED.
            1,0 IN PARPOS MEANS THE SECOND POSITIONAL PARAMETER CAN HAVE
        ONLY 1 ENTRY, BUT IT IS NOT REQUIRED.
            6 MEANS 'VOLSER' ENTERED CAN HAVE AT MOST 6 CHARACTERS.
            3,3 MEANS THE ENTERED VALUE FOR 'VOLSER' CAN CONTAIN ONLY
        ALPHABETIC, NATIONAL, OR NUMERIC CHARACTERS.
```

APPENDIX A

PROGRAM: PARSE CATEGORY #: Z

RESTRICTIONS: CONTINUED

(2) COMMAND: LISTALC STATUS SYSNAMES XKEYWD/YKEYWD (NONE REQUIRED.)

```
      CALL PARKEY(ISTAT)
      CALL PARNAM('/STATUS/')
      CALL PARKEY(ISYS)
      CALL PARNAM('/SYSNAMES/')
      CALL PARKEY(IKW)
      CALL PARNAM('/XKEYWD/')
      CALL PARNAM('/YKEYWD/')
      CALL PARSE(IRES)
```

NOTES: NO POSITIONAL PARAMETERS, AND ONLY KEYWORDS WITHOUT VALUE. HENCE NO CALLS TO PARDSN, PARDEC, OR PARPOS.
XKEYWD AND YKEYWD ARE MUTUALLY EXCLUSIVE. IKW IS SET TO 0 IF NEITHER ARE SPECIFIED, 1 IF XKEYWD IS SPECIFIED, OR 2 IF YKEYWD IS SPECIFIED.

(3) COMMAND: EDIT 'DSNAME' FORT/COBOL/ASM
LINESIZE('VALUE')/LRECL('VALUE')
NUM/NONUM

REQUIRED - 'DSNAME', A DATASET NAME.

DEFAULTS - FORT, NUM, 'VALUE'=80

```
      INTEGER*2 DSN(22),MEM(4),PASS(4),LEN(3)
      CALL PARDSN(DSN,MEM,PASS,LEN,1,1,
     * '/NAME OF DATASET TO BE EDITED./')
      CALL PARKEY(ITYPE,'/FORT/')
      CALL PARNAM('/FORT/')
      CALL PARNAM('/COBOL/')
      CALL PARNAM('/ASM/')
      CALL PARKEY(ILINE)
      CALL PARNAM('/LINESIZE/',14,'/LRECL/')
      CALL PARKEY(NUM,'/NUM/')
      CALL PARNAM('/NUM/')
      CALL PARNAM('/NONUM/')
      CALL PARSUB(14)
      CALL PARDEC(LINE,1)
      LINE=80
      CALL PARSE(IR)
```

NOTES: DEFINITION OF SUBFIELD 14 FOLLOWS THE DEFINITION OF ALL MAIN COMMAND PARAMETERS. 14 WAS ARBITRARY.
'/LRECL/' IS DEFINED AS AN ALIAS OF '/LINESIZE/'. SINCE ITS VALUE IS NUMERIC, PARDEC IS CALLED RATHER THAN PARPOS. BECAUSE IT IS NOT REQUIRED, ITS VALUE IS DEFAULTED TO 80. PARSE DOES NOT CHANGE THE NUMERIC FIELD, IF THE KEYWORD IS NOT USED. IF PARPOS WAS USED, DEFAULTS MUST BE SPECIFED AFTER PARSE IS CALLED, SUCH AS IF (ILINE.EQ.0) LINE=80

REFERENCES/ACKNOWLEDGEMENTS:
NONE

PROGRAM: QUIT CATEGORY #: Z

CATEGORY: ALL OTHERS

TITLE:
TERMINATES A FORTRAN PROGRAM WRITTEN AS A COMMAND PROCESSOR, SUPPLIES A RETURN CODE FOR THE PROCESSOR OR OPTIONALLY TRANSMITS ALPHAMERIC INFORMATION BACK TO A CALLING COMMAND.

TYPE: S SOURCE: AS DATE LAST CHG: 800805

AUTHOR: BILL JACOB, IBM, MODIFIED BY WALTER C. BODYCOMB, IBM

PURPOSE:
TO CONFORM TO TSO RETURN CODE STANDARDS AND MAKE THESE AVAILABLE IN COMMANDS BEING ACCESSED IN CLISTS, WITH &LASTCC. ALTERNATIVELY, QUIT CAN SUPPLY THE RETURN CODE TO A CALLING COMMAND AND ALSO TRANSMIT CHARACTER INFORMATION BACK TO THE COMMAND.

CALLING SEQUENCE:
CALL QUIT(IRC)
CALL QUIT(IRC,ARGLEN,ARGDAT)

ARGUMENTS:

NAME	TYPE	DIM	DESCRIPTION
IRC	INT*4	NO	INPUT VARIABLE WHICH SUPPLIES THE DESIRED RETURN CODE AFTER THE TERMINATION OF THE PROGRAM. IF 'ARGLEN' = 0 OR IS NOT INCLUDED IN THE CALL, 'IRC' IS ALSO ACCESSABLE BY THE &LASTCC VARIABLE OF A CALLING CLIST. IF 'ARGLEN' > 0, THE RETURN CODE IS NOT ACCESSABLE WITH &LASTCC IN A CLIST.
ARGLEN	INT*4	NO	THE LENGTH OF 'ARGDAT' IN BYTES. IF 'ARGLEN' = 0, THE CALL IS EQUIVALENT TO CALL QUIT(IRC).
ARGDAT	CHAR	ARGLEN BYTES	THE CHARACTER DATA TO BE TRANSMITTED BACK TO A CALLING COMMAND VIA THE SUBROUTINE "COMAND". IF 'ARGLEN' = 0, THE CALL IS EQUIVALENT TO CALL QUIT(IRC).

RESTRICTIONS:
1. SUBROUTINE CAN ONLY BE USED IN A MAIN PROGRAM EXECUTED VIA TSO AS A COMMAND OR BY THE "CALL" COMMAND.
2. 'ARGLEN' AND 'ARGDAT' ARE MEANINGFUL ONLY IF THIS ROUTINE IS IN A COMMAND THAT IS CALLED BY ANOTHER COMMAND BY SUBROUTINE 'COMAND'. IN THAT CASE, THE RETURN CODE 'IRC' IS NOT IN THE REGISTER ACCESSED BY &LASTCC OF A CLIST. TO PROVIDE A CORRECT RETURN CODE FOR CLIST USAGE AT ALL TIMES, SUPPLY A COMMAND WITH AN ALIAS TO BE USED ONLY WHEN IT IS CALLED FROM ANOTHER COMMAND, AND FORCE 'ARGLEN' = 0 IF THAT NAME IS NOT USED. THE NAME USED TO CALL A COMMAND IS AVAILABLE IN THE PROGRAM AS AN OPTION ARGUMENT IN SUBROUTINE PARSE.

REFERENCES/ACKNOWLEDGEMENTS:
NONE

APPENDIX A

PROGRAM: TSPTGT CATEGORY #: Z

CATEGORY: ALL OTHERS

TITLE:
WRITES MESSAGES TO AND GETS MESSAGES FROM A TERMINAL IN TSO COMMANDS, USING PUTLINE, GETLINE, OR PUTGET (IN PROMPT MODE).

TYPE: S SOURCE: AS DATE LAST CHG: 810127

AUTHOR: WALTER C. BODYCOMB, IBM

PURPOSE:
ALLOWS A WRITER OF TSO COMMANDS TO MAKE NEARLY FULL USE OF THE PUTLINE, GETLINE AND PUTGET FACILITIES. THIS INCLUDES:

(1) - INCLUDING MSGIDS SUBJECT TO THE MSGID/NOMSGID CHARACTERISTICS OF THE UADS PROFILE.
(2) - WRITE MULTILEVEL MESSAGES, EACH OF WHICH WILL BE FOLLOWED BY A PLUS ("+"). A HIGHER LEVEL MESSAGE IS DISPLAYED IF THE USER RESPONDS WITH "?".
(3) - FOLD INPUT AND OUTPUT TO UPPER CASE OR LEAVE AS IS.
(4) - WRITE OR RECEIVE MESSAGES SUBJECT TO A MSG/NOMSG AND PROMPT/NOPROMPT CLIST CONTROL ENVIRONMENT.
(5) - INHIBIT PRINTING THE RESPONSE OF A USER PUTGET PROMPT.
(6) - INHIBIT A CARRIAGE RETURN AFTER PRINTING A MESSAGE.
(7) - SOUND A BEEP, SUBJECT TO MSG/NOMSG CLIST CONTROL OR NOT.

CALLING PROCEDURE:

```
CALL TSPTGT(IRC,IEC,IOP,ARG...)              (PUTLINE)
CALL TSPTGT(IRC,IEC,IOP,AREA,NLEN)           (GETLINE)
CALL TSPTGT(IRC,IEC,IOP,AREA,NLEN,ARG...)    (PUTGET)
CALL TSPTGT(IRC,IEC)                         (PURGE, FORCE, OR BEEP)
```

ARGUMENTS:

NAME	TYPE	DIM	DESCRIPTION
IRC	INT	NO	OUTPUT RETURN CODE AS FOLLOWS:
			-6 INVALID PARAMETERS PASSED TO TSO GETLINE/PUTLINE/PUTGET (ERROR IN TSPTGT).
			-5 INVALID PARAMETERS PASSED TO TSPTGT: A. INVALID ENTRY CODE. B. NO OUTPUT MESSAGE WITH PUTLINE OR PUTGET. C. MORE THAN 2 MESSAGE LEVELS WITH A MULTI-LEVEL PUTLINE (WITHOUT THE DATA OPTION). D. MSGIDS SPECIFIED BUT THE FIRST LEVEL MESSAGE HAS NONE.
			-4 INSUFFICIENT MAIN STORAGE AVAILABLE.
			-3 THE MESSAGE LENGTH FOR A MESSAGE OR DATA LEVEL EXCEEDED 254 CHARACTERS.
			-2 TSPTGT WAS NOT CALLED FROM A TSO COMMAND.
			-1 GETLINE OR PUTGET ATTEMPTED AND THE IN-STORAGE LIST (NOT A COMMAND PROCEDURE) HAS BEEN EXHAUSTED. NLEN IS SET TO 0 AND NO DATA IS RETURNED.
			0 SUCCESSFUL COMPLETION.
			1 PUTGET OR GETLINE WITH TERM OPTION (IOP=32) WAS REQUESTED BUT:

PROGRAM: TSPTGT CATEGORY #: Z

ARGUMENTS: CONTINUED

A. NOPROMPT MODE IS CURRENTLY ACTIVE IN THE USERS ENVIRONMENT (PUTGET ONLY).
B. THE CURRENT SOURCE OF INPUT IS AN IN-STORAGE LIST (NOT A COMMAND PROCEDURE).
C. THE COMMAND IS RUNNING UNDER THE BACKGROUND TMP IN A COMMAND PROCEDURE BUT NO DATA PROMPT...ENDDATA GROUP WAS PROVIDED.

2 THE SIZE OF THE INPUT BUFFER SPECIFIED IN 'NLEN' IS NOT BIG ENOUGH TO HOLD THE ENTIRE MESSAGE ENTERED AT THE TERMINAL. THE INPUT IS TRUNCATED TO THE VALUE SPECIFIED BY NLEN AT THE ENTRY TO TSPTGT, AND NLEN IS SET TO THE SIZE OF THE MESSAGE BEFORE TRUNCATION.

3 THE USER SIGNALLED ATTENTION.

4 THE TERMINAL HAS BEEN DISCONNECTED.

IEC INT NO FACILITY REQUEST CODE AS FOLLOWS:

1 PUTLINE REQUEST. SEND A SINGLE OR TWO-LEVEL MESSAGE, OR A SINGLE OR MULTILINE DATA LINE TO A TERMINAL.

2 GETLINE REQUEST. GET A LINE OF INPUT FROM A TERMINAL (OR IN-LINE CLIST DATA...ENDDATA OR DATA PROMPT...ENDDATA GROUP).

3 PUTGET (PROMPT) REQUEST. SEND A SINGLE OR MULTI-LEVEL MESSAGE TO A TERMINAL AND GET A LINE IN RESPONSE FROM THE TERMINAL (OR IN-LINE CLIST DATA PROMPT...ENDDATA GROUP).

4 PURGE REQUEST. DELETE ALL EXISTING SECOND AND HIGHER LEVEL MESSAGES.

5 FORCE REQUEST. PRINT ALL EXISTING SECOND AND HIGHER LEVEL MESSAGES.

6 BEEP REQUEST. SOUND THE ALARM ON 3277 TYPE TERMINALS. THIS REQUEST (OR SERIES OF REQUESTS) SHOULD BE FOLLOWED BY A PUTLINE OR PUTGET. THIS REQUEST IS NOT AFFECTED BY CLIST MSG/NOMSG.

IOP INT NO THE OPTION CODE. THE CODE IS FORMED BY ADDING THE DESIRED VALUES FROM THE FOLLOWING LIST (IF NONE ARE DESIRED, IOP MUST BE ZERO):

1 MSGIDS ARE INCLUDED (E.G., IEF247I FOLLOWED BY A BLANK).

2 THIS IS A MULTILEVEL MESSAGE (THE USER CAN ENTER ? AND GET MORE INFORMATION), OR MULTILINE DATA.

4 FOLD OUTPUT TO UPPERCASE.

8 FOLD INPUT TO UPPERCASE.

16 PUTGET - NOPRINT OPTION. INHIBIT PRINTING WHEN THE USER ENTERS REPLY.
PUTLINE - DATA OPTION. OUTPUT LINES ARE DATA LINES, NOT MESSAGES. OUTPUT IS UNAFFECTED BY MSGID/NOMSGID, MSG/NOMSG, OR 2-LEVEL MAXIMUM (IEC=-5). MULTIPLE LINES ARE PRINTED IMMEDIATELY.

32 GETLINE - TERM OPTION. THIS CAUSES GETLINE TO RESPOND TO COMMAND PROCEDURE DATA PROMPT GROUPS RATHER THAN TO DATA GROUPS. IT IS NOT AFFECTED BY PROFILE OR CLIST CONTROL PROMPT/NOPROMPT.
PUTLINE/PUTGET - NORETURN OPTION. NO CARRIAGE

APPENDIX A

PROGRAM: TSPTGT CATEGORY #: Z

ARGUMENTS: CONTINUED

RETURN WILL BE DONE AT THE END OF THE OUTPUT LINE(S).

64 BEEP OPTION. SOUND ALARM ON 3277 TYPE TERMINALS. THIS OPTION MAY BE USED WITH PUTLINE OR PUTGET. IF USED WITHIN A CLIST WITH CONTROL NOMSG, THE ALARM WILL NOT BE SOUNDED (SEE NOTE 10).

128 NOATTENTION OPTION, USED WITH GETLINE AND PUTGET. NO ATTENTION EXIT WILL BE SET UP. IF THE USER SIGNALS ATTENTION, IT WILL BE TREATED AS A "LINE DELETE". (SEE NOTE 11.)

AREA CHAR YES A CONTIGUOUS AREA INTO WHICH THE DATA IS TO BE READ FROM THE TERMINAL. IF THE AMOUNT OF DATA IS > 0 BUT < THE LENGTH OF THE AREA, THE REMAINING SPACE IS FILLED WITH BLANKS.

NLEN INT NO THE NUMBER OF BYTES IN AREA. AFTER A SUCCESSFUL READ, NLEN CONTAINS THE LENGTH OF THE DATA READ FROM THE TERMINAL.

ARG CHAR OR INT AS NEEDED CONSISTS OF ZERO OR MORE ARGUMENTS WHICH SPECIFY THE MESSAGE TO BE SENT. SEVERAL FORMATS, WHICH CAN BE SINGLY OR IN COMBINATION, ARE ALLOWED:

1. A 'SELF-DEFINING' CHARACTER STRING. THE FIRST CHARACTER OF THE STRING IS TAKEN TO BE A DELIMITER AND TSPTGT FINDS THE END OF THE STRING BY SEARCHING FOR THE NEXT OCCURANCE OF THE SAME CHARACTER (WHICH SHOULD BE SOME PRINTABLE CHARACTER). THE STRING THUS FOUND IS ADDED TO THE MESSAGE. A NULL STRING (NO CHARACTERS BETWEEN THE DELIMITERS) CAUSES TSPTGT TO STOP EXAMINING THE ARGUMENTS IN THE CALL STATEMENT AND IMMEDIATELY PRINT THE MESSAGE. REMAINING ARGUMENTS, IF ANY, ARE IGNORED.
2. A 'DOUBLE ARGUMENT' CHARACTER STRING. TWO ARGUMENTS ARE USED. THE FIRST IS A POSITIVE FULLWORD INTEGER GIVING THE NUMBER OF CHARACTERS IN THE STRING. THE SECOND IS THE CHARACTER STRING WHICH IS TO BE ADDED TO THE MESSAGE. POSITIVE INTEGER ARGUMENTS WHICH ARE NOT FOLLOWED IMMEDIATELY BY CHARACTER ARGUMENTS ARE IGNORED.
3. FULLWORD INTEGER EQUAL TO ZERO. CAUSES TSPTGT TO STOP EXAMINING THE ARGUMENTS IN THE CALL STATEMENT AND IMMEDIATELY PRINT THE MESSAGE. REMAINING ARGUMENTS, IF ANY, ARE IGNORED.
4. FULLWORD INTEGER EQUAL TO NEGATIVE ONE. CAUSES TSPTGT TO STOP EXAMINING THE ARGUMENTS IN THE CALL STATEMENT AND TO 'HOLD' THE CURRENT CONTENTS OF THE MESSAGE. ON A SUBSEQUENT CALL TO TSPTGT THE MESSAGE MAY BE ADDED TO, OR PRINTED, OR BOTH. ONCE THE NEGATIVE ONE IS FOUND, THE REMAINING ARGUMENTS IN THE CALL STATEMENT, IF ANY, ARE IGNORED.
5. IF NO ARGUMENTS OTHER THAN THE RETURN CODE, ENTRY CODE, OPTION CODE (AND AREA AND NLEN FOR PUTGET)

PROGRAM: TSPTGT CATEGORY #: Z

ARGUMENTS: CONTINUED
VARIABLES ARE PROVIDED TO TSPTGT, IT PRINTS THE CURRENT CONTENTS OF THE BUFFER.

RESTRICTIONS:
SPECIFIED BY THE ERROR RETURN CODE, IRC.

REFERENCES/ACKNOWLEDGEMENTS:
IBM MANUAL GC28-0648, "OS/VS2 TSO GUIDE TO WRITING A TERMINAL MONITOR PROGRAM OR A COMMAND PROCESSOR." HOWEVER, THE FOLLOWING NOTES ON THE USE OF TSPTGT MAY BE HELPFUL:

1. TSO GETLINE, PUTLINE, AND PUTGET RESPOND TO THE TSO ENVIRONMENT AS SET UP BY THE PROFILE COMMAND (MSGID/NOMSGID, PROMPT/NOPROMPT), CLIST CONTROL (MSG/NOMSG, PROMPT/NOPROMPT), AND CLIST IN-LINE DATA AND PROMPT DATA GROUPS:
   - GETLINE - (WITHOUT TERM OPTION) CONTROLLED BY CLIST DATA..ENDDATA. WHEN USED BY A COMMAND IN A COMMAND PROCEDURE, THE COMMAND MUST BE FOLLOWED BY A DATA...ENDDATA GROUP. (WITH TERM OPTION, IOP=32) CONTROLLED BY CLIST DATA PROMPT...ENDDATA.
   - PUTLINE - (WITHOUT THE DATA OPTION) CONTROLLED BY PROFILE MSGID/NOMSGID AND CLIST MSG/NOMSG.
   - PUTGET - CONTROLLED BY PROFILE PROMPT/NOPROMPT, MSGID/NOMSGID AND CLIST MSG/NOMSG, PROMPT/NOPROMPT, DATA PROMPT... ENDDATA.
2. IF MULTILEVEL MESSAGES OR DATA LINES ARE SPECIFIED (IOP=2) THEN EACH OCCURRENCE OF ARG (OR 'DOUBLE ARG') IS CONSIDERED TO BE A NEW LEVEL OF MESSAGE. THIS INCLUDES THE USE OF MULTIPLE CALLS (-1 AS THE LAST ARGUMENT) TO BUILD MULTIPLE MESSAGE LEVELS.
3. MSGIDS MAY BE ANY LENGTH, MUST HAVE NO IMBEDDED BLANKS, AND MUST BE FOLLOWED BY A BLANK. MSGIDS ARE SPECIFIED ON THE FIRST-LEVEL MESSAGES, BUT NOT ON SUBSEQUENT LEVELS. MSGIDS ARE NOT SPECIFIED WITH PUTLINE WITH THE DATA OPTION (IOP=16).
4. MULTIPLE PUTGET CALLS USED TO BUILD A SINGLE MESSAGE OR SET OF MESSAGE LEVELS MUST INCLUDE AREA AND NLEN ON EACH CALL, EVEN THOUGH DATA WILL BE RETURNED ONLY ON THE LAST CALL. (NLEN WILL NOT BE CHANGED UNTIL THE LAST CALL.)
5. PUTLINE (WITHOUT THE DATA OPTION) CAN HAVE ONLY TWO LEVELS OF MESSAGE, WHILE PUTLINE DATA (IOP=16) AND PUTGET CAN HAVE MULTIPLE LEVELS.
6. BY TSO CONVENTION, ALL MULTILEVEL MESSAGES EXCEPT THE LAST END WITH A "+". TSPTGT WILL INSERT THE + AUTOMATICALLY UNLESS THE CALLER HAS ALREADY PROVIDED IT (MULTILINE DATA LINES DO NOT HAVE A +.)
7. COMMANDS USING GETLINE AND PUTGET IN A COMMAND PROCEDURE WITH DATA OR DATA PROMPT GROUPS MUST BE ABLE TO RECOGNIZE THE LAST LINE OF INPUT SINCE THERE IS NO RETURN CODE FOR "END OF DATA". IF THE END OF DATA NOT RECOGNIZED, THE NEXT PUTGET OR GETLINE (WITH THE TERM OPTION (IOP=32)) WILL GO TO THE TERMINAL, WHILE THE NEXT GETLINE WITHOUT THE TERM OPTION WILL GET THE NEXT LINE IN THE COMMAND PROCEDURE.

APPENDIX A

PROGRAM: TSPTGT CATEGORY #: Z

REFERENCES/ACKNOWLEDGEMENTS:

8. SECOND LEVEL MESSAGES ISSUED BY PUTLINE (WITHOUT THE DATA OPTION) ARE CHAINED BY TSO AND KEPT THROUGHOUT THE EXECUTION OF A COMMAND OR SUBCOMMAND UNLESS PRINTED (FORCE REQUEST) OR PURGED (PURGE REQUEST). COMMAND PROCESSORS USING PUTLINE (WITHOUT DATA OPTION) SECOND LEVEL MESSAGES SHOULD BE CAPABLE OF RECOGNIZING A "?" FROM THE USER AND RESPONDING WITH A TSPTGT FORCE CALL (BUT SEE NOTE 9).
9. ON ALL GETLINE REQUESTS, TSPTGT WILL CHECK THE INPUT FOR "?" BEFORE RETURNING. IF A ? (ONLY) IS FOUND, TSPTGT WILL ISSUE A FORCE (TO PRINT ANY EXISTING SECOND LEVEL MESSAGES) AND THEN REISSUE THE GETLINE.
10. THE BEEP OPTION (IOP=64) HAS VARYING EFFECTS, AS FOLLOWS:
    PUTLINE WITH THE DATA OPTION (IOP=16) ALWAYS SOUNDS THE TERMINAL ALARM AND PRINTS THE MESSAGE(S).
    PUTGET AND PUTLINE WITHOUT THE DATA OPTION WILL NOT SOUND THE ALARM (OR PRINT THE MESSAGE(S)) IF USED IN A CLIST WITH CONTROL NOMSG ACTIVE.
11. NORMALLY THE LAST ATTENTION EXIT SET UP IS THE ONLY EXIT WHICH WILL GET CONTROL IF THE USER SIGNALS A SINGLE ATTENTION. IN THE FOLLOWING SEQUENCE, TSPTGT WILL RECEIVE CONTROL AND RETURN THE ATTENTION CODE (IRC=3) TO THE CALLER:
    COMMAND CALLS ATTEN (SETTING UP ATTENTION EXIT.)
    COMMAND CALLS TSPTGT (GETLINE REQUEST.)
    (TSPTGT SETS UP THE ATTENTION EXIT AND WAITS FOR INPUT.)
    USER SIGNALS ATTENTION.
    IF THE CALLER OF TSPTGT WISHES HIS OWN ATTENTION EXIT TO GET CONTROL, HE MUST SPECIFY IOP=128. AN EXCEPTION TO THE NORMAL SCHEDULING OF ATTENTION EXITS OCCURS WHEN RUNNING IN A CLIST WITH AN ACTIVE ATTN GROUP. AN ACTIVE ATTN GROUP TAKES PRECEDENCE OVER ALL ATTENTION EXITS.
12. AN UNRECOGNIZED RETURN CODE FROM TSO PUTLINE, GETLINE, OR PUTGET WILL CAUSE A USER 100 ABEND WITH THE DUMP OPTION.

Appendix B.—Example of Command SYSOUT

```
C THIS LISTING ILLUSTRATES THE PRINCIPLES OF WRITING COMMANDS IN FORTRAN
      SUBROUTINE SYSOUT
      IMPLICIT INTEGER(A-Z)
C
C SYSOUT IS A COMMAND TO MAKE SYSOUT ALLOCATIONS ON TSO. THE SYNTAX IS:
C    SYSOUT   'DDNAME'   CLASS('CLASS')   DEST('RMTDEST')
C             USING('ATTRNAME')   COPIES('NUMBER')   CLOSE   REUSE
C             FORM('FORMNAME')   HOLD/NOHOLD
C WHERE
C     'DDNAME' - IS THE DDNAME ASSIGNED TO THE ALLOCATION (REQUIRED).
C     CLASS('CLASS') - DEFINES THE SYSOUT CLASS OF THE ALLOCATION. THE
C                DEFAULT IS A.
C     DEST('RMTDEST') - SPECIFIES THE REMOTE WORKSTATION TO PROCESS THE
C                SYSOUT FILE. THE DEFAULT IS THE USER'S DEFAULT
C                DESTINATION.
C     USING('ATTRNAME') - SPECIFIES A DUMMY FILE, SUCH AS MAY BE SET UP
C                WITH THE ATTRIBUTE COMMAND, TO COPY THE DCB FROM. THE
C                DEFAULT ASSIGNS NO DCB.
C     COPIES('NUMBER') - PERMITS THE NUMBER OF COPIES TO BE SPECIFIED
C                (FROM 1 TO 255). THE DEFAULT IS 1.
C     CLOSE     - SPECIFIES THAT THE FILE SHOULD BE UNALLOCATED WHEN IT
C                IS CLOSED. THE DEFAULT REQUIRES THE USER TO
C                UNALLOCATE IT. (THE FILE MUST BE UNALLOCATED BEFORE IT
C                IS PROCESSED.)
C     REUSE     - SPECIFIES THAT 'DDNAME' SHOULD BE FREED IF IT IS
C                ALREADY ALLOCATED.
C     FORM('FORMNAME') - PERMITS THE USER TO SPECIFY A FORM TO BE USED.
C     HOLD      - PLACES THE FILE IN THE HOLD QUEUE INSTEAD OF
C                PROCESSING IT.
C     NOHOLD    - PROCESSES THE FILE WHEN IT IS UNALLOCATED. THIS IS THE
C                DEFAULT IF HOLD IS NOT USED.
C
C THE RETURN CODES ARE
C            0 - ALLOCATION SUCCESSFUL.
C          1XX - PARSE FAILED WITH ERROR XX.
C        > 500 - ALLOCATION FAILED. THE VALUE IS THE ERROR REASON CODE.
C TYPICAL REASONS ARE
C                1040 - 'DDNAME' IS IN USE.
C                1056 - 'DDNAME' IS OPEN AND CANNOT BE FREED.
C                1108 - 'ATTRNAME' CANNOT BE FOUND.
C                1132 - 'RMTDEST' IS NOT A LEGAL REMOTE STATION NAME.
C
CREATE THE DYNAMIC ALLOCATION TEXTS AND OTHER VARIABLES.
      INTEGER*2   DD(7)/1,1,5*0/,   SYS(4)/24,1,1,0/,   A/'A'/,
     * DEST(7)/88,1,5*0/,    HOLD(2)/89,0/,    DCB(7)/45,1,5*0/,
     * CLOSE(2)/28,0/,   COPY(4)/29,1,1,0/,    FORM(5)/26,1,3*0/,
     * PERM(2)/82,0/,    UNALLC(2)/7,0/,       COPIES(2)
      EQUIVALENCE (COPIES(1),COP)
C
COMMENCE THE PARSE.
C
CALL FOR REQUIRED POSITIONAL PARAMETER, DDNAME.
      CALL PARPOS(DD(4),LDD,'/DDNAME/',1,17,8,1,3,'/DDNAME/')
CALLS FOR KEYWORD "CLASS".
      CALL PARKEY(ICLASS)
      CALL PARNAM('/CLASS/'.1)
CALLS FOR KEYWORD "COPIES" ("COPY" ALSO ACCEPTED).
```

APPENDIX B

```
      CALL PARKEY(INFO)
      CALL PARNAM('/COPIES/',2,'/COPY/')
CALLS FOR KEYWORD "DEST".
      CALL PARKEY(IDEST)
      CALL PARNAM('/DEST/',3)
CALLS FOR KEYWORDS "HOLD" AND "NOHOLD" WITH DEFAULT OF "NOHOLD".
      CALL PARKEY(IHOLD,'/NOHOLD/')
      CALL PARNAM('/HOLD/')
      CALL PARNAM('/NOHOLD/')
CALLS FOR KEYWORD "CLOSE".
      CALL PARKEY(ICLOSE)
      CALL PARNAM('/CLOSE/')
CALLS FOR KEYWORD "USING".
      CALL PARKEY(IUSING)
      CALL PARNAM('/USING/',4)
CALLS FOR KEYWORD "REUSE".
      CALL PARKEY(IREUSE)
      CALL PARNAM('/REUSE/')
CALLS FOR KEYWORD "FORM".
      CALL PARKEY(IFORM)
      CALL PARNAM('/FORM/',5)
CALLS TO SET THE VALUE OF "CLASS" AS SPECIFIED BY THE USER.
      CALL PARSUB(1)
      CALL PARPOS(SYS(4),IERR '/SYSOUT CLASS/',1,1,1,5,5,
     * '/SYSOUT CLASS/')
CALLS TO SET THE VALUE OF "COPIES" AS SPECIFIED BY THE USER.
      CALL PARSUB(2)
      CALL PARDEC(COP,1)
COP DEFAULTS TO 1 IF THE USER DOES NOT SPECIFY THIS KEYWORD.
      COP=1
CALLS TO SET THE VALUE OF "DEST" AS SPECIFIED BY THE USER.
      CALL PARSUB(3)
      CALL PARPOS(DEST(4),LDEST,'/DEST/',1,17,8,3,3,'/DEST/')
CALLS TO SET THE VALUE OF "DCB" AS SPECIFIED BY THE USER.
      CALL PARSUB(4)
      CALL PARPOS(DCB(4),LDCB,'/ATTRNAME/',1,17,8,1,3,'/ATTRNAME/')
CALLS TO SET THE VALUE OF "FORM" AS SPECIFIED BY THE USER.
      CALL PARSUB(5)
      CALL PARPOS(FORM(4),FORM(3),'/FORM NAME/',1,1,4,3,3,'/FORM NAME/')
CALL TO SET ALL THE VALUES ENTERED BY THE USER.
      CALL PARSE(IER)
CHECK IF PARSE WAS SUCCESSFUL.
      IF (IER.EQ.0) GOTO 10
CHANGE RETURN CODE FOR UNSUCCESSFUL PARSE.
      LC=100+IER
CONVERT TO CHARACTER FORM SURROUNDED BY SLASHES.
      CALL TSINT(IER,DD)
CONVEY FAILURE TO USER (WITH BEEP).
      CALL TSPTGT(IER,1,64,'/SYSOUT PARSE FAILED WITH ERROR /',DD,'/./')
CONCLUDE COMMAND WITH RETURN CODE LC.
    5 CALL QUIT(LC)
C
CORRECT LENGTH TEXTS FOR DYNALC.
   10 DD(3)=LDD
      DEST(3)=LDEST
      DCB(3)=LDCB
C
CHECK FOR REUSE.
```

```
      IF (IREUSE.EQ.0) GOTO 15
CALL TO UNALLOCATE THE DDNAME.
      CALL DYNALC(2,IER,LC,INFO,0,DD,UNALLC)
CONTINUE IF FREED OR IF DDNAME WAS NOT ALLOCATED (ERROR 1080).
      IF (IER.EQ.0.OR.LC.EQ.1080) GOTO 15
COMMUNICATE ERROR.
      CALL DYNALE
      GOTO 5
C
CORRECTIONS FOR OTHER KEYWORDS.
C
CHECK ON USING KEYWORD. DISABLE TEXT IF KEYWORD IS NOT SPECIFIED.
   15 IF (IUSING.EQ.0) DCB(1)=0
CHECK ON DEST KEYWORD. DISABLE TEXT IF KEYWORD IS NOT SPECIFIED.
      IF (IDEST.EQ.0) DEST(1)=0
CHECK ON CLASS KEYWORD. DEFAULT TO CLASS(A) IF KEYWORD IS NOT SPECIFIED.
      IF (ICLASS.EQ.0) SYS(4)=A
CHECK ON CLOSE KEYWORD. DISABLE TEXT IF KEYWORD IS NOT SPECIFIED.
      IF (ICLOSE.EQ.0) CLOSE(1)=0
CHECK ON COPIES KEYWORD. ADJUST LIMITS.
      IF (COP.LT.1) COP=1
      IF (COP.GT.255) COP=255
COPY LAST BYTE OF COP TO COPY(4). FIRST MOVE BINARY VALUE ONE BYTE LEFT.
      COP=COP*256
      COPY(4)=COPIES(2)
CHECK ON HOLD KEYWORD. DISABLE TEXT IF NOHOLD DEFAULTED OR SPECIFIED.
      IF (IHOLD.EQ.2) HOLD(1)=0
CHECK ON FORM KEYWORD. DISABLE TEXT IF KEYWORD IS NOT SPECIFIED.
      IF (IFORM.EQ.0) FORM(1)=0
C
CAREFULLY ALLOCATE THE SYSOUT FILE. RETURN CODE SET TO REASON CODE.
      CALL DYNALC(1,IER,LC,INFO,0,DD,SYS,DEST,HOLD,DCB,CLOSE,COPY,FORM,
     * PERM)
C
CONCLUDE IF ALLOCATION WAS OK.  ERROR MESSAGE IF NOT.
      IF (IER.NE.0) CALL DYNALE
      GOTO 5
      END
```

Appendix C.—Example of Command READN

```
      SUBROUTINE READN
      IMPLICIT INTEGER (A-Z)
C READN IS A COMMAND FOR CLIST USE ONLY THAT PROMPTS FOR A SINGLE LINE
C OF INPUT OF AT MOST 72 CHARACTERS, SUPPRESSING PRINTING ON IBM 3270-
C TYPE TERMINALS. A CLIST VARIABLE IS SET TO THE ENTERED STRING. THE
C ENTIRE PROCESS CAN BE PRECEDED BY UP TO 5 LEVELS OF PROMPTING
C MESSAGES. THE SYNTAX IS:
C    READN 'CLISTVARNAME' ''MSG1'' ''MSG2'' ''MSG3'' ''MSG4'' ''MSG5''
C          NORETURN/RETURN
C WHERE
C  'CLISTVARNAME' IS THE NAME OF THE CLIST VARIABLE TO BE SET TO THE
C                 INPUT STRING. IT MUST BE PRESET IN THE CLIST TO A
C                 STRING WITH AT LEAST AS MANY CHARACTERS AS EXPECTED
C                 IN RESPONSE TO THE PROMPT. IT MUST ALSO NOT BE A
C                 GLOBAL VARIABLE OR A SYSTEM VARIABLE.
C  ''MSG1'' AND ALL OTHER MESSAGES ARE PROMPTING MESSAGES. IF ''MSG2''
C                 IS USED, ''MSG1'' WILL END WITH A "+", AND ''MSG2''
C                 WILL BE DISPLAYED ONLY IF THE USER ENTERS "?". THE
C                 SAME APPLIES TO OTHER MESSAGES. IF "?" IS NOT ENTERED
C                 THE REMAINING MESSAGES ARE DISCARDED. EACH MESSAGE IS
C                 LIMITED TO 254 CHARACTERS.
C  RETURN         RETURNS THE CURSER TO A NEW LINE FOR THE RESPONSE TO
C                 THE PROMPT. THIS IS THE DEFAULT IF NEITHER NORETURN
C                 NOR RETURN IS SPECIFIED.
C  NORETURN       REQUESTS THE CURSER TO REMAIN ON THE SAME LINE AS THE
C                 PROMPTING MESSAGE.
C
C THE CLIST USING THIS COMMAND MUST HAVE
C       CONTROL PROMPT
C IN EFFECT. ANY ERROR MESSAGE IS DISPLAYED ONLY IF
C       CONTROL MSG
C IS IN EFFECT.
C
C THE RETURN CODES ARE:
C              0, SUCCESSFUL PUTGET AND CLIST VARIABLE SETTING.
C              1, INPUT WAS TRUNCATED TO 72 CHARACTERS AND THE RESULT
C                 WAS SUCCESSFULLY TRANSMITTED TO 'CLISTVARNAME'.
C              4, CALLER WAS NOT IN A CLIST.
C              6, 'CLISTVARNAME' WAS NOT PREDEFINED IN THE CLIST.
C              7, 'CLISTVARNAME' WAS NOT A CORRECT VARIABLE TYPE.
C              8, 'CLISTVARNAME' WAS NOT PREDEFINED WITH A LONG ENOUGH
C                 STRING.
C             12, CONTROL PROMPT WAS NOT IN EFFECT IN THE CLIST.
C             20, THE USER HIT ATTENTION AS THE PROMPT INPUT.
C             24, THERE IS A BUG IN THE TSPTGT SUBROUTINE. PLEASE CALL
C                 PROGRAMMING ASSISTANCE.
C             28, THE TERMINAL WAS DISCONNECTED.
C            1XX, THE PARSE FAILED WITH ERROR XX. SEE SUBROUTINE PARSE.
C
CREATE DIMENSIONED VARIABLES AND INITIALIZE.
      INTEGER*4 LMSG(5),I/'TEST'/
      LOGICAL*1 MSG(254,5),VAR(31),PWD(72),FLAG/.FALSE./,TMSG(72)
C
CHECK FOR CLIST USE.
      CALL GETCSV(I,4,MSG,LMSG,LC)
      IF (LC.EQ.4) GOTO 40
C
```

APPENDIX C

```
COMMENCE THE PARSE.
CLIST VARIABLE NAME TO BE SET IS A POSITIONAL PARAMETER.
      CALL PARPOS(VAR,LVAR,'/CLIST VARIABLE NAME/',1,17,31,4,5,
     * '/CLIST VARIABLE NAME/')
CALL TO DEFINE THE FIRST STRING TO BE WRITTEN TO THE TERMINAL.
      CALL PARSTR(MSG,LMSG,254,18,'/'' ''/')
CALLS TO ALLOW FOUR ADDITIONAL LEVELS OF MESSAGES.
      DO 2 I=2,5
    2 CALL PARSTR(MSG(1,I),LMSG(I),254,16)
CALLS FOR RETURN/NORETURN KEYWORDS (RETURN DEFAULTED).
      CALL PARKEY(IRET,'/RETURN/')
      CALL PARNAM('/RETURN/')
      CALL PARNAM('/NORETURN/')
CONCLUDE PARSING.
      CALL PARSE(IER)
CHECK FOR SUCCESSFUL PARSE.  SEND ERROR MESSAGES IF IT FAILS.
      IF (IER.EQ.0) GOTO 10
      LC=100+IER
      CALL TSINT(IER,MSG)
      CALL TSPTGT(IER,1,64,'/READN PARSE ERROR = /',MSG,'/./')
CONCLUDE COMMAND WITH RETURN CODE.
    5 CALL QUIT(LC)
C
CHECK IF VAR IS IN THE CALLER'S CLIST PROPERLY.
   10 CALL GETCSV(VAR,LVAR,TMSG,LTMSG,LC)
      GOTO (12,12,12,40,5,45,50),LC
   12 LPWD=72
CORRECT IOP FOR RETURN/NORETURN KEYWORDS.
      LC=94
      IF (IRET.EQ.2) LC=126
C
CALL TO PUTGET FACILITY TO COMMUNICATE MESSAGES AND AWAIT FOR REPLY.
C
      CALL TSPTGT(IER,3,LC,PWD,LPWD,LMSG,MSG,LMSG(2),MSG(1,2),
     * LMSG(3),MSG(1,3),LMSG(4),MSG(1,4),LMSG(5),MSG(1,5))
CHECK IF PUTGET WAS OK.
      IF (IER) 15,35,20
COMMUNICATE ACCURATE ERROR MESSAGES IF PUTGET FAILS.
   15 LC=24
      CALL TSPTGT(IER,1,64,'/TSPTGT ERROR IN READN - PLEASE CALL PROGRAM
     *MING ASSISTANCE./')
      GOTO 5
   20 GOTO (21,23,24,25), IER
   21 LC=12
      CALL TSPTGT(IER,1,64,'/READN NOT USED IN PROMPT ENVIRONMENT./')
      GOTO 5
CHANGE FLAG TO INDICATE TRUNCATION FOR LATER RETURN CODE.
   23 FLAG=.TRUE.
      LPWD=72
      CALL TSPTGT(IER,1,64,'/READN INPUT TRUNCATED TO 72 CHARACTERS./')
      GOTO 35
   24 LC=20
      GOTO 5
   25 LC=28
      GOTO 5
C
CHANGE THE CLIST VARIABLE VALUE TO THE STRING ENTERED.
C
```

```
   35 CALL SETCSV(VAR,LVAR,PWD,LPWD,LC)
      IF (LC.EQ.0) GOTO 38
COMMUNICATE ACCURATE ERROR MESSAGES IF SETCSV FAILS.
      GOTO (5,5,5,40,5,45,50,55,5), LC
CORRECT RETURN CODE IF TRUNCATION HAD OCCURRED.
   38 IF (FLAG) LC=1
      GOTO 5
   40 CALL TSPTGT(IER,1,64,'/READN USEABLE ONLY IN A CLIST./')
      GOTO 5
   45 CALL TSPTGT(IER,1,64,'/READN ERROR - /',
     * LVAR,VAR,'/ IS NOT A CLIST VARIABLE./')
      GOTO 5
   50 CALL TSPTGT(IER,1,64,'/READN ERROR - /',
     * LVAR,VAR,'/ IS AN INCORRECT VARIABLE TYPE./')
      GOTO 5
   55 CALL TSINT(LTMSG,MSG)
      CALL TSPTGT(IER,1,64,'/READN ERROR - INPUT CANNOT EXCEED /',
     * MSG,'/ CHARACTERS./')
      GOTO 5
      END
```

Appendix D.—Example of Command Compress

```
      SUBROUTINE CMPRSS
      IMPLICIT INTEGER (A-Z)
COMPRESS COMPRESSES A PDS IN THE FOREGROUND USING THE UTILITY SPFCOPY
C WHICH IS FUNCTIONALLY EQUIVALENT TO IEBCOPY. ITS SYNTAX IS:
C          COMPRESS   'PDSNAME'   VOLUME('VOLSER')   OLD/RELEASE/SHR
C                     NOPRINT/PRINT
C WHERE
C   'PDSNAME' - IS THE NAME OF THE PDS TO BE COMPRESSED.
C   VOLUME('VOLSER') - SPECIFIES THE VOLUME ON WHICH 'PDSNAME' RESIDES.
C                IF NOT SPECIFIED, A CATALOG SEARCH IS MADE.
C   OLD        - ALLOCATES 'PDSNAME' OLD AND DOES NOT RELEASE EXTRA
C                SPACE. THIS IS THE DEFAULT IF NONE OF OLD, RELEASE, OR
C                SHR IS SPECIFIED.
C   RELEASE    - ALLOCATES 'PDSNAME' MOD AND RELEASE EXTRA SPACE AFTER
C                IT IS COMPRESSED, TO THE NEAREST TRACK.
C   SHR        - ALLOCATES 'PDSNAME' SHR AND DOES NOT RELEASE EXTRA
C                SPACE. EXTREME CAUTION SHOULD BE EXERCISED WITH THIS
C                KEYWORD BY INSURING OTHER USERS ARE NOT ALSO ALLOCATING
C                SHR AND WRITING INTO THE DATASET. IT IS THE USER'S
C                RESPONSIBILITY TO DO THIS.
C   PRINT      - REQUESTS THAT THE SPFCOPY SYSPRINT DATASET BE PRINTED
C                AT THE TERMINAL.
C   NOPRINT    - REQUESTS THAT THE SPFCOPY SYSPRINT DATASET BE DUMMIED
C                OUT. THIS IS THE DEFAULT IF PRINT IS NOT SPECIFIED.
C THE RETURN CODES ARE
C          20 - THE USER SIGNALLED ATTENTION, OR
C               'PDSNAME' WAS NOT FOUND, OR
C               'VOLSER' WAS NOT MOUNTED, OR
C               'PDSNAME' WAS SPECIFIED ONLY WITH A MEMBER NAME, OR
C               'PDSNAME' WAS NOT A PARTITIONED DATASET, OR
C               'PDSNAME' COULD NOT BE CHECKED, OR
C               'PDSNAME' WAS BUSY.
C          ALL OTHER RETURN CODES (0, 4, OR 8) ARE FROM SPFCOPY.
C
COMMON FOR ATTR TO MAKE AVAILABLE DCB DATA FOR 'PDSNAME' AFTER THE
CALL TO RECFM (VOLUME IS IN LLL(45) AND DSORG IN LLL(86)).
      COMMON/ATTR/ LLL(96)
COMMON FOR ATTN TO PREVENT SHIFTING OF ATTENTION HANDLING CHECKS.
      COMMON/ATTN/ ATTENT
C
C   STR IS EQUIVALENCED TO STR8 AND WILL CONTAIN SYSIN DATA.
      LOGICAL*1 LLL, ATTENT, STR(40)
C
C  LEN CONTAINS THE 'PDSNAME' LENGTHS.
      INTEGER*4 LEN(3)
C
C  INTEGER*2 CONTAINS MOSTLY TEXTS FOR ALLOCATION.
      INTEGER*2 DS(25)/2,1,0,22*' '/, DD(7)/85,1,8,4*' '/,DDS(8),
     * DISP(4)/4,1,1,Z0100/, MOD/Z0200/, SHR/Z0800/, RLSE(2)/13,0/,
     * PRIM1(5)/10,1,3,0,256/, PASS(7)/80,1,0,4*' '/, PERM(2)/82,0/,
     * CONV(2)/83,0/, VOL(7)/16,1,6,4*' '/, TRK(2)/7,0/, DUMMY(2)/36,0/,
     * SEC19(5)/11,1,3,0,4864/, SYSDA(6)/21,1,5,'SY','SD','A'/,
     * TERM(2)/40,0/, PO/512/, PRIMR(5)/10,1,3,0,256/, TRKR(2)/7,0/,
     * PRM(4), PARMS/88/, V3340(6)/21,1,5,'V3','34','0'/
C
C ALTNAM IS THE ALTERNATE DDNAME LIST PASSED TO SPFCOPY. SYSIN, SYSPRT,
C SYSUT3, AND SYSUT4 WILL BE THE SYSTEM-GENERATED DDNAMES IN SPFCOPY.
```

APPENDIX D

```
C STR8 CONTAINS THE SYSIN DATA. NM PROVIDES SPACE FOR THE NUMBER OF
C BYTES OF ATNAME.
      REAL*8 NM(12),ALTNAM(11)/11*Z0000000000000000/,SYSIN,SYSPRT, MEM,
     * SYSUT3,SYSUT4,STR8(5)/' COPY IN','DD=','    ',OUTD','D=',' '/,
     * DDNAME
C
C DDS SET UP TO INSURE DDNAME IS ON A DOUBLE WORD BOUNDARY.
C PARMS IS A HALF WORD VARIABLE PRECEDING THE ALTNAM LIST. NM AND PRM
C   SET UP TO INSURE PARMS IS ON A HALF WORD BOUNDARY.
C THE FOLLOWING EQUIVALENCES MAKE IT EASY TO REFERENCE THE FOUR FILES
C   REQUIRED BY SPFCOPY.
      EQUIVALENCE (DD(4),DDNAME),
     * (NM(2),ALTNAM(1)),(ALTNAM(5),SYSIN),(ALTNAM(6),SYSPRT),
     * (ALTNAM(10),SYSUT3),(ALTNAM(11),SYSUT4),(PRM(1),NM(1)),
     * (PRM(4),PARMS),(STR(1),STR8(1)),(DDS(2),DD(1))
C
COMMENCE PARSE.
C
CALL TO GET 'PDSNAME', A REQUIRED POSITIONAL PARAMETER.
      CALL PARDSN(DS(4),MEM,PASS(4),LEN,1,17,'/PDS NAME/')
CALLS FOR VOLUME KEYWORD.
      CALL PARKEY(IVOL)
      CALL PARNAM('/VOLUME/',1)
CALLS FOR KEYWORDS OLD, RELEASE (RLSE ACCEPTED), AND SHR.
      CALL PARKEY(IREL,'/OLD/')
      CALL PARNAM('/OLD/')
      CALL PARNAM('/RELEASE/','/RLSE/')
      CALL PARNAM('/SHR/')
CALLS FOR KEYWORDS PRINT AND NOPRINT.
      CALL PARKEY(IPRINT)
      CALL PARNAM('/NOPRINT/')
      CALL PARNAM('/PRINT/')
CALLS TO SET VOLUME VALUE IF SPECIFIED.
      CALL PARSUB(1)
      CALL PARPOS(VOL(4),IERR,'/VOLUME/',1,17,6,3,3,'/VOLUME/')
CALL TO PARSE TO FINALIZE EVERY VARIABLE.
      CALL PARSE(IER)
CONCLUSION OF PARSE.
C
CHECK IF PARSE WAS OK. COMMUNICATE MESSAGE IF NOT.
      IF (IER.EQ.0) GOTO 10
      LC=100+IER
      CALL TSINT(IER,DS)
      CALL TSPUT(64,'/COMPRESS PARSE FAILED WITH ERROR = /',DS,'/./')
CONCLUDE COMMAND WITH RETURN CODE.
    5 CALL QUIT(LC)
CONSTRUCT ATTENTION HANDLER.
   10 ATTENT=.FALSE.
      CALL ATTEN(ATTENT)
CHECK IF ONLY A MEMBER NAME WAS SPECIFIED. WRITE ERROR MESSAGE IF SO.
      IF (LEN(1)) 20,15,25
   15 CALL TSPUT(64,'/NO DATASET TO COMPRESS./')
   17 LC=20
      GOTO 5
CORRECT FOR NEGATIVE DSNAME LENGTH (ENTERED QUOTED).
   20 LEN(1)=-LEN(1)
CANCEL THIS COMMAND IF 'SYS1.LPALIB' IS THE DSNAME.
   25 IF (LEN(1).EQ.11.AND.LSTCM(DS(4),'SYS1.LPALIB',11).EQ.0) GOTO 17
```

```
CHECK EXISTANCE AND PDS.
      CALL RECFM(DS(4),LEN(1),VOL(4),IER)
      IF (IER.LT.300) GOTO 30
COMMUNICATE MESSAGE IF THERE IS A BAD IER.
      CALL RECMSG
      GOTO 17
CHECK IF DATASET IS A PDS. COMMUNICATE MESSAGE IF NOT.
   30 IF (LSTCM(PO,LLL(86),2).EQ.0) GOTO 35
      CALL TSPUT(64,'/''/'.LEN(1),DS(4),
     * '/'' IS NOT A PARTITIONED DATASET./')
      GOTO 17
CHANGE PASS KEY FOR ENTERED PASSWORD WITH DATASET.
   35 IF (LEN(3).NE.0) PASS(3)=LEN(3)
      IF (LEN(3).EQ.0) PASS(1)=0
CHECK VOLUME KEYWORD. MOVE VOLUME FROM THE ATTR COMMON TO AVOID ANOTHER
CATALOG SEARCH FOR DYNAMIC ALLOCATION.
      IF (IVOL.EQ.0) CALL CHMOV(LLL(45),VOL(4),6)
COMMUNICATE MESSAGE ABOUT MEMBER IF SPECIFIED.
      IF (LEN(2).NE.0) CALL TSPUT(64,'/MEMBER NAME /',LEN(2),MEM,
     * '/ IGNORED./')
CHANGE DISP, ETC, FOR RELEASE, SHR, OR OLD.
      GOTO (40,45,50), IREL
   50 DISP(4)=SHR
   40 RLSE(1)=0
      PRIMR(1)=0
      TRKR(1)=0
      GOTO 55
   45 DISP(4)=MOD
CAREFULLY ALLOCATE THE DATASET. FIRST CORRECT THE DSNAME TEXT LENGTH.
   55 DS(3)=LEN(1)
      CALL DYNALC(1,IER,IERR,INFO,0,DD,DS,CONV,DISP,RLSE,PRIMR,TRKR,
     * PASS,VOL)
CHECK IF ALLOCATION WAS OK. IF NOT SEND MESSAGE.
      IF (IER.EQ.0) GOTO 60
      CALL DYNALE
      GOTO 17
   60 IF (ATTENT) GOTO 70
CHANGE SYSIN LINE TO CONTAIN THE SYSTEM-GENERATED DDNAME.
      CALL CHMOV(DDNAME,STR(12),8)
      CALL CHMOV(DDNAME,STR(27),8)
CAREFULLY ALLOCATE SYSIN TO A VIO DATASET, NO NAME NEEDED.
      CALL DYNALC(1,IER,IERR,INFO,0,DD,V3340,PRIM1,TRK,CONV)
CHECK ON SYSIN ALLOCATION. ERROR MESSAGE IF BAD.
      IF (IER.NE.0) GOTO 62
CORRECT THE ALTERNATE DDNAME LIST FOR SYSIN.
      SYSIN=DDNAME
CREATE THE SYSIN FILE BY WRITING THE SINGLE LINE.
      CALL TSPOPN(IER,1,DDNAME)
      CALL TSPPUT(IER,1,34,STR)
      CALL TSPCLS(IER,1)
CAREFULLY ALLOCATE SYSUT3. WRITE ERROR MESSAGES IF BAD.
      CALL DYNALC(1,IER,IERR,INFO,0,DD,SYSDA,PRIM1,SEC19,TRK,CONV)
      IF (IER.NE.0) GOTO 62
CORRECT THE ALTERNATE DDNAME LIST FOR SYSUT3.
      SYSUT3=DDNAME
CAREFULLY ALLOCATE SYSUT4. WRITE ERROR MESSAGES IF BAD.
      CALL DYNALC(1,IER,IERR,INFO,0,DD,SYSDA,PRIM1,SEC19,TRK,CONV)
      IF (IER.EQ.0) GOTO 63
```

APPENDIX D

```
   62 CALL DYNALE
      GOTO 70
CORRECT THE ALTERNATE DDNAME LIST FOR SYSUT4.
   63 SYSUT4=DDNAME
CONSIDER SYSPRINT. NO CHECK ON ERROR BECAUSE ERROR RARELY MADE.
      IF (IPRINT.LE.1) CALL DYNALC(1,IER,IERR,INFO,0,DD,DUMMY,CONV,PERM)
      IF (IPRINT.EQ.2) CALL DYNALC(1,IER,IERR,INFO,0,DD,TERM,CONV,PERM)
CORRECT THE ALTERNATE DDNAME LIST FOR SYSPRINT.
      SYSPRT=DDNAME
CHECK ATTENTION FLAG.
      IF (ATTENT) GOTO 70
COMMUNICATE WARNING TO USER.
      CALL TSPUT(0,'/COMPRESS STARTED. DO NOT SIGNAL ATTENTION OR THE DA
     *TASET MAY BE LOST./')
COMMENCE EXECUTION OF SPFCOPY. THE ATTENTION ARGUMENT SET TO IGNORE.
      CALL ATTACH(1,LC,'SPFCOPY ',0,PARMS)
CHECK TO SEE IF ATTENTION WAS HIT OR IF PRINT KEYWORD USED. GO TO THE
CLEANUP SECTION IF SO.
      IF (IPRINT.EQ.2.OR.LC.EQ.-1) GOTO 70
CHECK IF COMPRESS WAS OK. MESSAGE IF SO.
      IF (LC.NE.0) GOTO 64
      CALL TSPUT(0,'/''/',LEN(1),DS(4),'/'' SUCCESSFULLY COMPRESSED./')
      GOTO 70
CHECK IF AN ABEND OCCURRED. MESSAGE IF NOT.
   64 IF (LC.LT.0) GOTO 65
      CALL TSINT(LC,DD)
      CALL TSPUT(64,'/''/',LEN(1),DS(4),
     * '/'' COMPRESS ENDED WITH CONDITION CODE = /',DD,'/./')
COMPRESS ABENDED. EXTRACT HEX CODE FROM RETURN ARGUMENT.
   65 CALL TSHEX(-LC,3,2,DD)
      CALL TSPUT(64,'/SYSTEM HEX ABEND CODE FOR COMPRESS = /',DD,'/./')
CLEANUP. FREE DATASET IF SHR WAS NOT USED.
   70 IF (IREL.NE.3) CALL DYNALC(2,IER,IERR,INFO,0,DS,TRK)
CORRECT RETURN CODE FOR ABEND OR ATTENTION.
      IF (ATTENT.OR.LC.LT.0) GOTO 17
      GOTO 5
      END
```

DISTRIBUTION

ADMINISTRATOR
DEFENSE TECHNICAL INFORMATION CENTER
ATTN DTIC-DDA (12 COPIES)
CAMERON STATION, BUILDING 5
ALEXANDRIA, VA 22314

COMMANDER
US ARMY MATERIEL DEVELOPMENT &
READINESS COMMAND
5001 EISENHOWER AVENUE
ALEXANDRIA, VA 22333

DIRECTOR
US ARMY BALLISTIC RESEARCH LABORATORY
ATTN DRDAR-TSB-S (STINFO)
ABERDEEN PROVING GROUND, MD 21005

USAMSSA
THE PENTAGON, ROOM DD1034
WASHINGTON, DC 20310

CACI
ATTN MAILLARK, LOC 02-14
1815 NORTH FORT MYER DR
ARLINGTON, VA 22209

COMMANDER
US ARMY COMMUNICATIONS & ELECTRONICS
MATERIEL READINESS COMMAND
ATTN COMPUTER CENTER
FT MONMOUTH, NJ 07703

COMMANDER
US ARMY COMMUNICATIONS RESEARCH &
DEVELOPMENT COMMAND
ATTN COMPUTER CENTER
ATTN DRDCO-TCS, CENTER FOR TACTICAL
COMPUTER SYS
FT MONMUTH, NJ 07703

US ARMY COMPUTER SYSTEMS COMMAND
BUILDING 1465
FT BELVOIR, VA 22060

COMMANDER
US ARMY COMPUTER SYSTEMS COMMAND
ATTN COMPUTER CENTER
ATTN TECH LIBRARY
FT BELVOIR, VA 22060

US ARMY ELECTRONICS TECHNOLOGY
& DEVICES LABORATORY
ATTN DELET-DD
FT MONMOUTH, NJ 07703

ALMASA
AUTOMATED LOGISTICS MANAGEMENT
SYSTEMS ACTIVITY
ST LOUIS, MO 63188

DIRECTOR
US ARMY MATERIEL SYSTEMS ANALYSIS
ACTIVITY
ATTN DRXSY-MP
ABERDEEN PROVING GROUND, MD 21005

COMMANDER
US ARMY MISSILE & MUNITIONS
CENTER & SCHOOL
ATTN ATSK-CTD-F
REDSTONE ARSENAL, AL 35809

US ARMY MOBILITY EQUIPMENT RESEARCH &
DEVELOPMENT COMMAND
ATTN DRDME-E
ATTN DRDME-EC (2 COPIES)
ATTN DRDME-EA (2 COPIES)
ATTN DRDME-EM (2 COPIES)
ATTN DRDME-EE (2 COPIES)
ATTN DRCPM-MEP-D (2 COPIES)
ATTN DRCPM-MEP-M (2 COPIES)
ATTN DRCPM-MEP-T (2 COPIES)
ATTN ATZA-TSM-G (2 COPIES)
FT BELVOIR, VA 22060

DIRECTOR
NIGHT VISION & ELECTRO-OPTICS
LABORATORY
ATTN COMPUTER CENTER
FT BELVOIR, VA 22060

COMMANDER
US ARMY RSCH & STD GP (EUR)
ATTN CHIEF, PHYSICS & MATH BRANCH
FPO NEW YORK 09510

COMMANDER
US ARMY TRAINING & DOCTRINE COMMAND
ATTN ATCD-DCS, COMBAT DEVELOMENT
ATTN ATCD-AN, ANALYSIS DIR
ATTN ATCD-AN, COMBAT SYS BR
ATTN ATCD-AN, METHODOLOGY BR
ATTN ATCD-C, TELECOM COMD & CON &
COMPTR SYS DIR
ATTN ATCD-C, BATTLEFIELD SYS
INTEGRATION BR
ATTN ATCD-IE, INTEL & EW DIR
ATTN ATCD-IE, ELECTRONIC WRFARE BR
ATTN ATCD-T, TEST & EVAL DIR
ATTN DCS COMBAT DEVELOPMENTS--ATCD
ATTN DCS TRAINING-ATTING
ATTN INFO OFFICE--ATID
ATTN COMM & ELECTRONICS--ATCE
FT MONROE, VA 23651

COMMANDER
NAVAL AVIATION LOGISTICS CENTER
CODE 04 B
ATTN STEVE BERNSTEIN
NAVAL AIR STATION
PATUXENT RIVER, MD 20670

COMMANDING OFFICER
NAVAL TRAINING EQUIPMENT CENTER
ATTN TECHNICAL LIBRARY
ORLANDO, FL 32813

COMMANDER
HQ AF DATA AUTOMATION AGENCY
ATTN DATA SYS EVAL OFFICE
GUNTER AFB, AL 36114

AFDSC/CC
AIR FORCE DATA SERVICES CENTER
THE PENTAGON
WASHINGTON, DC 20330

HQ, USAF/SAMI
WASHINGTON, DC 20330

DEPARTMENT OF COMMERCE
NATIONAL BUREAU OF STANDARDS
ATTN HOWARD BLOOM
ATTN CARL WENGER
ATTN CITA FURLANI
ATTN THEODORE HOPP
GAITHERSBURG, MD 20760

DEPARTMENT OF ENERGY
ATTN B. H. AUDET
ATTN MICHAEL BOBLETT
GERMANTOWN, MD 20545

AMES RESEARCH CENTER
NASA
ATTN TECHNICAL INFO DIV
MOFFET FIELD, CA 94035

DIRECTOR
NASA
GODDARD SPACE FLIGHT CENTER
ATTN 250, TECH INFO DIV
GREENBELT, MD 20771

DIRECTOR
NASA
LANGLEY RESEARCH CENTER
ATTN TECHNICAL LIBRARY
HAMPTON, VA 23665

ENGINEERING SOCIETIES LIBRARY
ATTN ACQUISITIONS DEPARTMENT
345 EAST 47TH STREET
NEW YORK, NY 10017

JACOB & SUNDSTROM, INC
SUITE 2214
ATTN WILLIAM C. JACOB, JR.
THE WORLD TRADE CENTER BALTIMORE
BALTIMORE, MD 21202

TELEDYNE BROWN ENGINEERING
CUMMINGS RESEARCH PARK
ATTN DR. MELVIN L. PRICE, MS-44
HUNTSVILLE, AL 35807

TEXAS INSTRUMENTS, INC
PO BOX 226015
ATTN FRANK POBLENZ
DALLAS, TX 75266

US ARMY ELECTRONICS RESEARCH
& DEVELOPMENT COMMAND
ATTN COMMANDER, DRDEL-CG
ATTN TECHNICAL DIRECTOR, DRDEL-CT
ATTN PUBLIC AFFAIRS OFFICE, DRDEL-IN

HARRY DIAMOND LABORATORIES
ATTN CO/TD/TSO/DIVISION DIRECTORS
ATTN RECORD COPY, 81200
ATTN HDL LIBRARY, 81100 (3 COPIES)
ATTN HDL LIBRARY, 81100 (WOODBRIDGE)
ATTN TECHNICAL REPORTS BRANCH, 81300
ATTN LEGAL OFFICE, 97000
ATTN CHAIRMAN, EDITORIAL COMMITTEE
ATTN ZABLUDOWSKI, B., 47400 (GIDEP)
ATTN CHIEF, 21000
ATTN CHIEF, 21100
ATTN CHIEF, 21200
ATTN CHIEF, 21300
ATTN CHIEF, 21400
ATTN CHIEF, 21500
ATTN CHIEF, 22000
ATTN CHIEF, 22100
ATTN CHIEF, 22300
ATTN CHIEF, 22800
ATTN CHIEF, 22900
ATTN CHIEF, 20240
ATTN CHOY, S., 48000
ATTN ROSEN, R., 48000
ATTN BLACKWELL, A., 00251
ATTN BUCKHEIT, S., 00251
ATTN HINE, R., 00251
ATTN KIRCHER, K., 00251
ATTN MATTHEWS, J., 00251
ATTN SLEDGE, J., 00251
ATTN VOSS, K., 00251
ATTN WILLIAMS, J., 00251
ATTN HAUSNER, A., 00251 (10 COPIES)

END

FILMED

1-84

DTIC